

A Logic for True Concurrency

Paolo Baldan and Silvia Crafa

Department of Pure and Applied Math, University of Padova

Abstract. We propose a logic for true concurrency whose formulae predicate about events in computations and their causal dependencies. The induced logical equivalence is hereditary history preserving bisimilarity, and fragments of the logic can be identified which correspond to other true concurrent behavioural equivalences in the literature: step, pom-set and history preserving bisimilarity. Standard Hennessy-Milner logic, and thus (interleaving) bisimilarity, is also recovered as a fragment. We also propose an extension of the logic with fixpoint operators, thus allowing to describe causal and concurrency properties of infinite computations. We believe that this work contributes to a rational presentation of the true concurrent spectrum and to a deeper understanding of the relations between the involved behavioural equivalences.

1 Introduction

In the semantics of concurrent and distributed systems, a major dichotomy opposes the *interleaving approaches*, where concurrency of actions is reduced to the non-deterministic choice among their possible sequentializations, to *true-concurrent approaches*, where concurrency is taken as a primitive notion. In both cases, on top of the operational models a number of behavioural equivalences have been defined by abstracting from aspects which are considered unobservable [vG01,vGG01].

For the interleaving world, a systematic and impressive picture is taken in the linear-time branching-time spectrum [vG01]. Quite interestingly, the equivalences in the spectrum can be uniformly characterised in logical terms. Bisimilarity, the finest equivalence, corresponds to Hennessy-Milner (HM) logic: two processes are bisimilar if and only if they satisfy the same HM logic formulae [HM85]. Coarser equivalences correspond to suitable fragments of HM logic.

In the true-concurrent world, relying on models like event structures or transition systems with independence [WN95], several behavioural equivalences have been defined, ranging from hereditary history preserving (hhp-) bisimilarity, to pomset and step bisimilarity. Correspondingly, a number of logics have been studied, but, to the best of our knowledge, a unifying logical framework encompassing the main true-concurrent equivalences is still missing. The huge amount of work on the topic makes it impossible to give a complete account of related approaches. Just to give a few references (see Section 7 for a wider discussion), [DNF90] proposes a general framework encompassing a number of temporal and modal logics that characterise pomset and weak history preserving bisimilarities as well as interleaving bisimilarity. However, finer equivalences are not considered and a single unitary logic is missing. History preserving (hp-) bisimilarity has been characterised in automata-theoretic terms using HD-automata [MP97] or Petri nets [Vog91]. Concerning hhp-bisimilarity, several logics with modalities corresponding to the “retraction” or “backward” execution of computations have been proposed [NC95,Bed91,HS85,PU11]. In absence of autoconcurrency they are shown to capture hhp-bisimilarity, while the general case complicates the picture and requires some adjustments.

In this paper we propose a behavioural logic for concurrency and we show that it allows to characterise a relevant part of the truly concurrent spectrum. More specifically, the full logic \mathcal{L} is shown to capture hhp-bisimilarity, the finest equivalence in the spectrum in [vGG01]. Then suitable fragments of the logic are shown to scale down to the characterisation of other coarser equivalences, i.e., history preserving, pomset and step bisimilarity. Standard HM logic, and thus (interleaving) bisimilarity, is also recovered as a fragment.

Our logic allows to predicate about events in computations together with their causal and independence relations. It is interpreted over prime event structures, but it could naturally be

interpreted over any formalism with a notion of event, causality and consistency. A formula is evaluated in a configuration representing the current state of the computation, and it predicates on the possible future evolutions starting from that state. The logic is event-based in the sense that it contains an operator acting as a binder: it asserts the existence of an event satisfying suitable requirements and it bind the event to a variable so that the event can be referred later in the formula. In this respect, it is reminiscent of the modal analogue of independence-friendly modal logic as considered in [BF02].

Specifically, the logic contains two main operators. The formula $(x, \overline{y} < \mathbf{a} z)\varphi$ declares that an \mathbf{a} -labelled future event exists, which causally depends on the event bound to x , and is independent from the event bound to y . Such an event is bound to variable z so that it can be later referred to in φ . In general, x and y can be replaced by tuples of variables. A second operator allows to “execute” events previously bound to variables. The formula $\langle z \rangle \varphi$ says that the event bound to z is enabled in the current state, and after its execution φ holds.

Different behavioural equivalences are induced by fragments of the logics where we suitably restrict the set of possible futures the formulae are able to refer to. Namely, hhp-bisimilarity, that is captured by the full logic, corresponds to the ability of observing the existence of a number of legal but (possibly) incompatible futures. Such ability is strictly related to the capability of observing future events without executing them (in fact the execution of an event would rule out all the events in conflict with it). Interestingly, the definition of hhp-bisimilarity is normally given in terms of backward transitions, whereas our logical characterisation has a “forward flavor”. By restricting to a fragment where future events can be observed only by executing them (any occurrence of the binding operator is immediately followed by a corresponding execution), we get hp-bisimilarity. Pomset bisimilarity is induced by a fragment of the logic obtained by further restricting that for hp-bisimilarity, with the requirement that propositional connectives are used only on closed (sub)formulae. Roughly speaking, this fragment predicates about the possibility of executing pomset transitions and the closedness requirement prevents pomset transitions from being causally linked to the events in the past. Finally, quite intuitively, step bisimilarity corresponds to the possibility of observing only currently enabled concurrent actions.

The logic \mathcal{L} in its basic form is essentially a means to understand and compare the different process equivalences, but its expressive power is rather weak. In fact, although events arbitrarily far in the future can be “observed”, the logic only allows to describe computations where a finite number of events are executed. In order to overcome this limitation and to provide a more powerful specification logic, well-suited for describing properties of unbounded, possibly infinite computations, we enrich the logic with a form of recursion. More specifically, we extend the logic by adding least (and dually greatest) fixpoint operators, thus obtaining a kind of first order modal mu-calculus in the style of [Dam96,DFG98] and of the fixpoint extension of Independence-Friendly Modal Logic in [BK05]. In the resulting logic \mathcal{L}_μ one can express non-trivial causal properties, like “any \mathbf{a} action can be always followed by a causally related \mathbf{b} action in at most three steps”, or “an \mathbf{a} action can be always executed in parallel with a \mathbf{b} action”. Moreover, we show that, as it happens in the interleaving case, the addition of the fixpoint operator does not alter the logical equivalence hence the logical equivalence of \mathcal{L}_μ is still hhp-bisimulation.

We believe that this work contributes to the definition of a logical counterpart of the true concurrent spectrum, shading further light on the relations between the involved behavioural equivalences and suggests interesting directions of investigations in the verification of true-concurrency properties.

The rest of the paper is organised as follows. In Section 2 we introduce the basics of event structures and the concurrent equivalences we will work with in the paper. In Section 3 we present the syntax and semantics of our logic. In Section 4 we study the logical equivalence induced by the logic, proving that it coincides with hhp-bisimulation. In Section 5 we provide a characterisation of others concurrent equivalences in terms of fragments of our logic. In Section 6 we discuss the fixpoint extension of our logic. Finally, in Section 7 we discuss some related work and present directions of future research.

2 Background

In this section we provide the basics of prime event structures which will be used as models for our logic. Then we define some common behavioural concurrent equivalences which will play a basic role in the paper.

2.1 Event structures

Prime event structures [Win87] are a widely known model of concurrency. They describe the behaviour of a system in terms of events and dependency relations between such events. Throughout the paper Λ denotes a fixed set of labels ranged over by $\mathbf{a}, \mathbf{b}, \mathbf{c} \dots$

Definition 1 (prime event structure). A (Λ -labelled) prime event structure (PES) is a tuple $\mathcal{E} = \langle E, \leq, \#, \lambda \rangle$, where E is a denumerable set of events, $\lambda : E \rightarrow \Lambda$ is a labelling function and $\leq, \#$ are binary relations on E , called causality and conflict respectively, such that:

1. \leq is a partial order and $\lceil e \rceil = \{e' \in E \mid e' \leq e\}$ is finite for all $e \in E$;
2. $\#$ is irreflexive, symmetric and hereditary with respect to \leq , i.e., for all $e, e', e'' \in E$, if $e \# e' \leq e''$ then $e \# e''$.

In the following, we will assume that the components of an event structure \mathcal{E} are named as in the definition above. Subscripts carry over the components.

Definition 2 (consistency, concurrency). Let \mathcal{E} be a PES. We say that $e, e' \in E$ are consistent, written $e \frown e'$, if $\neg(e \# e')$. We say that e and e' are concurrent, written $e \parallel e'$, if $\neg(e \leq e'), \neg(e' \leq e)$ and $\neg(e \# e')$.

Causality and concurrency will be sometimes used on set of events. Given $X \subseteq E$ and $e \in E$, by $X < e$ we mean that for all $e' \in X$, $e' < e$. Similarly $X \parallel e$, resp. $X \frown e$, means that for all $e' \in X$, $e' \parallel e$, resp. $e' \frown e$. We write $\lceil X \rceil$ for $\bigcup_{e \in X} \lceil e \rceil$.

The idea of (concurrent) computation is captured, in event structures, by the notion of configuration.

Definition 3 (configuration). Let \mathcal{E} be a PES. A (finite) configuration in \mathcal{E} is a (finite) pairwise consistent subset of events $C \subseteq E$ closed w.r.t. causality (i.e., $\lceil C \rceil = C$). The set of finite configurations of \mathcal{E} is denoted by $\mathcal{C}(\mathcal{E})$.

Observe that the empty set of events \emptyset is always a configurations, which can be intended as the initial state of the computation.

Hereafter, unless explicitly stated otherwise, all configurations will be assumed to be finite. A pairwise consistent subset $X \subseteq E$ of events will be always seen as a *pomset* (partially ordered multiset) (X, \leq_X, λ_X) , where \leq_X and λ_X are the restrictions of \leq and λ to X . Given $X, Y \subseteq E$ we will write $X \sim Y$ if X and Y are isomorphic as pomsets.

Definition 4 (pomset transition and step). Let \mathcal{E} be a PES and let $C \in \mathcal{C}(\mathcal{E})$. Given $\emptyset \neq X \subseteq E$, if $C \cap X = \emptyset$ and $C' = C \cup X \in \mathcal{C}(\mathcal{E})$ we write $C \xrightarrow{X} C'$ and call it a pomset transition from C to C' . When the events in X are pairwise concurrent, we say that $C \xrightarrow{X} C'$ is a step. When $X = \{e\}$ we write $C \xrightarrow{e} C'$ instead of $C \xrightarrow{\{e\}} C'$.

A PES \mathcal{E} is called *image finite* if for any $C \in \mathcal{C}(\mathcal{E})$ and $\mathbf{a} \in \Lambda$, the set of events $\{e \in E \mid C \xrightarrow{e} C' \wedge \lambda(e) = \mathbf{a}\}$ is finite. All the PESs considered in this paper will be assumed to be image finite. As it commonly happens when relating modal logics and bisimilarities, this assumption is crucial for getting a logical characterisation of the various bisimulation equivalences in Sections 4 and 5, based on a finitary logic.

2.2 Concurrent behavioural equivalences

Behavioural equivalences which capture to some extent the concurrency feature of a system, can be defined on the transition system where states are configurations and transitions are pomset transitions.

Definition 5 (pomset, step bisimulation). Let $\mathcal{E}_1, \mathcal{E}_2$ be PESS. A pomset bisimulation is a relation $R \subseteq \mathcal{C}(\mathcal{E}_1) \times \mathcal{C}(\mathcal{E}_2)$ such that if $(C_1, C_2) \in R$ and $C_1 \xrightarrow{X_1} C'_1$ then $C_2 \xrightarrow{X_2} C'_2$, with $X_1 \sim X_2$ and $(C'_1, C'_2) \in R$, and vice versa. We say that $\mathcal{E}_1, \mathcal{E}_2$ are pomset bisimilar, written $\mathcal{E}_1 \sim_p \mathcal{E}_2$, if there exists a pomset bisimulation R such that $(\emptyset, \emptyset) \in R$.

Step bisimulation is defined analogously, replacing general pomset transitions with steps. We write $\mathcal{E}_1 \sim_s \mathcal{E}_2$ when \mathcal{E}_1 and \mathcal{E}_2 are step bisimilar.

While pomset and step bisimilarity only consider the causal structure of the current step, (hereditary) history preserving bisimilarities are sensible to the way in which the executed events depend on events in the past. In order to define history preserving bisimilarities the following definition is helpful.

Definition 6 (posetal product). Given two PESS $\mathcal{E}_1, \mathcal{E}_2$, the posetal product of their configurations, denoted $\mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$, is defined as

$$\{(C_1, f, C_2) : C_1 \in \mathcal{C}(\mathcal{E}_1), C_2 \in \mathcal{C}(\mathcal{E}_2), f : C_1 \rightarrow C_2 \text{ isomorphism}\}$$

A subset $R \subseteq \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$ is called a posetal relation. We say that R is downward closed when for any $(C_1, f, C_2), (C'_1, f', C'_2) \in \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$, if $(C_1, f, C_2) \subseteq (C'_1, f', C'_2)$ pointwise and $(C'_1, f', C'_2) \in R$ then $(C_1, f, C_2) \in R$.

Given a function $f : X_1 \rightarrow X_2$ we will denote by $f[x_1 \mapsto x_2] : X_1 \cup \{x_1\} \rightarrow X_2 \cup \{x_2\}$ the function defined, for $z \in X_1 \cup \{x_1\}$, by

$$f[x_1 \mapsto x_2](z) = \begin{cases} x_2 & \text{if } z = x_1 \\ f(z) & \text{otherwise} \end{cases}$$

Note that the same notation can represent an update of f , when $x_1 \in X_1$, or an extension of its domain, otherwise.

Definition 7 ((hereditary) history preserving bisimulation). A history preserving (hp-)bisimulation is a posetal relation $R \subseteq \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$ such that if $(C_1, f, C_2) \in R$ and $C \xrightarrow{e_1} C'_1$ then $C_2 \xrightarrow{e_2} C'_2$, with $(C'_1, f[e_1 \mapsto e_2], C'_2) \in R$, and vice versa. We say that $\mathcal{E}_1, \mathcal{E}_2$ are history preserving (hp-)bisimilar and write $\mathcal{E}_1 \sim_{hp} \mathcal{E}_2$ if there exists a hp-bisimulation R such that $(\emptyset, \emptyset, \emptyset) \in R$.

A hereditary history preserving (hhp-)bisimulation is a downward closed hp-bisimulation. The fact that $\mathcal{E}_1, \mathcal{E}_2$ are hereditary history preserving (hhp-)bisimilar is denoted $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$.

It is easy to see ([vGG01]) that the definition of (h)hp-bisimilarity can be equivalently given by using pomset transitions instead of single event transitions, i.e., by asking that if $(C_1, f, C_2) \in R$ and $C \xrightarrow{X_1} C'_1$ then there exists $C_2 \xrightarrow{X_2} C'_2$ and $(C'_1, f', C'_2) \in R$, with $f'_{|C_1} = f$.

3 A logic for true concurrency

In this section we introduce the syntax and the semantics of our logic. The formulae are interpreted over PESS. They predicate about events in computations and their dependencies as primitive concepts.

In order to keep the notation simple, tuples of variables like x_1, \dots, x_n will be denoted by \mathbf{x} and, abusing the notation, tuples will be often used as sets.

Definition 8 (syntax). Let Var be a denumerable set of variables ranged over by x, y, z, \dots . The syntax of the logic \mathcal{L} over the set of labels Λ is defined as follows, where \mathbf{a} ranges over Λ :

$$\varphi ::= \top \mid \varphi \wedge \varphi \mid \neg \varphi \mid (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi \mid \langle z \rangle \varphi$$

The operator $(\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z)$ acts as binder for the variable z , as clarified by the following notion of free variables in a formula.

Definition 9 (free variables). The set of free variables of a formula φ , denoted $fv(\varphi)$, is inductively defined by:

$$\begin{aligned} fv(\top) &= \emptyset \\ fv(\varphi_1 \wedge \varphi_2) &= fv(\varphi_1) \cup fv(\varphi_2) \\ fv(\neg \varphi) &= fv(\varphi) \\ fv((\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi) &= (fv(\varphi) \setminus \{z\}) \cup \mathbf{x} \cup \mathbf{y} \\ fv(\langle z \rangle \varphi) &= fv(\varphi) \cup \{z\} \end{aligned}$$

The satisfaction of a formula φ is defined with respect to a configuration $C \in \mathcal{C}(\mathcal{E})$, representing the state of the computation, and a function $\eta : Var \rightarrow E$, called an *environment*, that binds free variables in φ to events in the future of C . In particular, the events bound to free variables in a formula must be both consistent with the actual state of the computation and pairwise consistent. Such a requirement is expressed by the following definition of legal pair.

Definition 10 (environments, legal pairs). Let \mathcal{E} be a PES. We denote by $Env_{\mathcal{E}}$ the set of environments, i.e., $Env_{\mathcal{E}} = Var \rightarrow E$. Given a formula φ in \mathcal{L} , a pair $(C, \eta) \in \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}$ is legal for φ if $C \cup \eta(fv(\varphi))$ is a set of pairwise consistent events. We denote by $lp_{\mathcal{E}}(\varphi)$ the set of legal pairs for φ .

We simply write Env and $lp(\varphi)$, omitting the subscript, when the PES \mathcal{E} is clear from the context. Moreover, in order to simplify the definition of the semantics, given a configuration C , we denote by $E[C]$ the *residual* of E after C , defined as $E[C] = \{e \mid e \in E \setminus C \wedge C \frown e\}$.

Definition 11 (semantics). Let \mathcal{E} be a PES. The denotation of a formula φ , written $\llbracket \varphi \rrbracket^{\mathcal{E}} \in 2^{\mathcal{C}(\mathcal{E}) \times Env}$ is defined inductively as follow:

$$\begin{aligned} \llbracket \top \rrbracket^{\mathcal{E}} &= \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \\ \llbracket \varphi_1 \wedge \varphi_2 \rrbracket^{\mathcal{E}} &= \llbracket \varphi_1 \rrbracket^{\mathcal{E}} \cap \llbracket \varphi_2 \rrbracket^{\mathcal{E}} \\ \llbracket \neg \varphi \rrbracket^{\mathcal{E}} &= lp(\varphi) \setminus \llbracket \varphi \rrbracket^{\mathcal{E}} \\ \llbracket (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi \rrbracket^{\mathcal{E}} &= \{(C, \eta) \mid (C, \eta) \in lp((\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi) \text{ and} \\ &\quad \exists e \in E[C] \text{ such that} \\ &\quad \lambda(e) = \mathbf{a} \wedge \eta(\mathbf{x}) < e \wedge \eta(\mathbf{y}) \parallel e \\ &\quad \wedge (C, \eta[z \mapsto e]) \in \llbracket \varphi \rrbracket^{\mathcal{E}}\} \\ \llbracket \langle z \rangle \varphi \rrbracket^{\mathcal{E}} &= \{(C, \eta) \mid C \xrightarrow{\eta(z)} C' \wedge (C', \eta) \in \llbracket \varphi \rrbracket^{\mathcal{E}}\} \end{aligned}$$

When $(C, \eta) \in \llbracket \varphi \rrbracket^{\mathcal{E}}$ we say that the PES \mathcal{E} satisfies the formula φ in the configuration C and environment $\eta : Var \rightarrow E$, and write $\mathcal{E}, C \models_{\eta} \varphi$. For closed formulae φ , we write $\mathcal{E} \models \varphi$, when $\mathcal{E}, \emptyset \models_{\emptyset} \varphi$.

Intuitively, the formula

$$(\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi$$

holds in (C, η) when in the future of the configuration C there is an \mathbf{a} -labelled event e , consistent with the events bound to free variables in φ such that binding e to variable z , the formula φ holds. Such an event is required to be caused (at least) by the events already bound to variables in \mathbf{x} ,

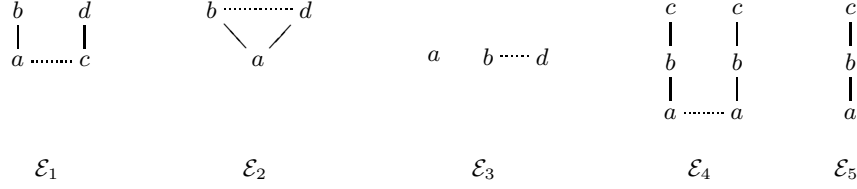


Fig. 1.

and to be independent (at least) from those bound to variables in \mathbf{y} . We stress that the event e might not be currently enabled; it is only required to be consistent with the current configuration, meaning that it could be enabled in the future of the current configuration. The formula $\langle z \rangle \varphi$ says that the event bound to z is currently enabled, hence it can be executed producing a new configuration which satisfies the formula φ . To simplify the notation we write $(\mathbf{a} z) \varphi$ for $(\langle \mathbf{a} z \rangle \varphi)$.

As an example, consider the PES \mathcal{E}_1 in Fig. 1, corresponding to the CCS process $a.b + c.d$, where dotted lines represent immediate conflict and the causal order proceeds upwards along the straight lines. The empty configuration satisfies the closed formula $(\mathbf{b} x)\mathbf{T}$, i.e., $\mathcal{E}_1 \models (\mathbf{b} x)\mathbf{T}$, even if the \mathbf{b} -labelled event is not immediately enabled. Also $\mathcal{E}_1 \models (\mathbf{b} x)\mathbf{T} \wedge (\mathbf{d} y)\mathbf{T}$, since there are two possible (incompatible) computations that starts from the empty configuration and contain, respectively, a \mathbf{b} -labelled and a \mathbf{d} -labelled event. On the other hand, if $\varphi = (\mathbf{a} z)\langle z \rangle ((\mathbf{b} x)\mathbf{T} \wedge (\mathbf{d} y)\mathbf{T})$ then $\mathcal{E}_1 \not\models \varphi$ since after the execution of the \mathbf{a} -labelled event, \mathcal{E}_1 reaches a configuration that does not admit a future containing an event labelled by \mathbf{d} . As a further example, the formula φ above is satisfied by the PESs \mathcal{E}_2 and \mathcal{E}_3 in Fig. 1 corresponding respectively to the process $a.(b + d)$ and $a \mid (b + d)$, whereas the formula $(\mathbf{a} z)\langle z \rangle (\bar{z} < \mathbf{b} x)\mathbf{T}$ is satisfied only by \mathcal{E}_3 .

It is worth noticing that the semantics of the binding operator does not prevent from choosing for z an event e that has been already bound to a different variable, i.e., the environment function η needs not to be injective. This is essential to avoid the direct observation of conflicts, a capability which would make the logical equivalence stricter than hhp-bisimilarity (and of any reasonable behavioural equivalence). Consider for instance the PESs associated to the hhp-equivalent processes $a + a$ and a : in order to be also logically equivalent, they both must satisfy the formula $(\mathbf{a} z)(\mathbf{a} z')\mathbf{T}$. Hence for the second PES, both z and z' must be bound to the unique \mathbf{a} -labelled event. On the other hand, observe that both PESs falsify the formula $(\mathbf{a} z)(\mathbf{a} z')\langle z \rangle \langle z' \rangle \mathbf{T}$. In fact, z' must be bound to an event consistent with that associated to z (because z occurs free in $\langle z \rangle \langle z' \rangle \mathbf{T}$). Hence z and z' will be bound to the same event, which cannot be executed twice.

3.1 About environments and legal pairs

It is immediate to see that, according to Definition 11, the denotation of a formula always consists of a set of legal pairs for the formula.

Lemma 1 (denotations consist of legal pairs). *Let \mathcal{E} be a PES. Then for any formula $\varphi \in \mathcal{L}$, it holds $\llbracket \varphi \rrbracket^{\mathcal{E}} \subseteq lp_{\mathcal{E}}(\varphi)$*

Proof. Straightforward induction on the structure of the formula φ . We only comment case $\varphi = \langle z \rangle \psi$. If $(C, \eta) \in \llbracket \varphi \rrbracket^{\mathcal{E}}$ then, by definition, if we let $e = \eta(z)$, it holds that $C \xrightarrow{e} C \cup \{e\}$ and $(C \cup \{e\}, \eta) \in \llbracket \psi \rrbracket^{\mathcal{E}}$. Hence by inductive hypothesis $(C \cup \{e\}, \eta) \in lp_{\mathcal{E}}(\psi)$, i.e., $C \cup \{e\} \cup \eta(fv(\psi))$ is pairwise consistent. Since $fv(\varphi) = fv(\psi) \cup \{z\}$, we have that $C \cup \eta(fv(\varphi)) = C \cup \{e\} \cup \eta(fv(\psi))$, and thus we can conclude $(C, \eta) \in lp_{\mathcal{E}}(\varphi)$. \square

Moreover, the semantics of a formula only depends on the events that the environment associates to the free variables of the formula.

Lemma 2. *Let \mathcal{E} be a PES and let $C \in \mathcal{C}(E)$. Let $\varphi \in \mathcal{L}$ and let $\eta_1, \eta_2 : \text{Var} \rightarrow E$ be environments such that $\eta_1(x) = \eta_2(x)$ for any $x \in fv(\varphi)$. Then*

$$\mathcal{E}, C \models_{\eta_1} \varphi \quad \text{iff} \quad \mathcal{E}, C \models_{\eta_2} \varphi$$

In particular, $(C, \eta_1) \in lp_{\mathcal{E}}(\varphi)$ if and only if $(C, \eta_2) \in lp_{\mathcal{E}}(\varphi)$.

Proof. Routine induction on the structure of φ . \square

Note that without restricting the semantics of formulae to legal pairs the logics would have been too powerful. Indeed, it would allow to observe conflicts through a combination of the binder and the execution modality. For instance, consider the PESS \mathcal{E}_4 and \mathcal{E}_5 in Fig. 1, corresponding to the processes $a.b.c + a.b.c$ and $a.b.c$ and take formula $\varphi = (a\ x)(b\ y)\langle x \rangle \neg \langle y \rangle \top$, saying that there are two events labelled by a and b such that after executing the first, the second cannot be executed. With the current definition neither \mathcal{E}_4 nor \mathcal{E}_5 satisfy φ , since after binding x to any a -labelled event e , in order to keep the denotation legal, y must be bound to the b -labelled event caused by e , that is executable after e . Without the restriction to legal pairs, instead, the formula would hold in \mathcal{E}_4 , since variables x and y could be bound to conflicting events (e.g., x could be bound to the a -labelled event on the left and y to the b -labelled event on the right). Similarly, consider the formula $\psi = (a\ x)(b\ y)\neg(x, y < c\ z)\top$, saying that there are two events, labelled by a and b , respectively, which are not common causes for any c -labelled event. Also ψ does not hold either in \mathcal{E}_4 nor in \mathcal{E}_5 . Omitting the restriction to legal pairs, ψ would be true only in \mathcal{E}_4 where x and y can be bound to conflicting events. This means that the logic would allow one to distinguish the PESS corresponding to any process from that corresponding to the non-deterministic choice between that process and itself, which instead are equated by virtually any behavioural equivalence.

Instead of restricting the semantics to legal pairs, one could envisage syntactic constraints which produce essentially the same effect, thus limiting the observation power of the logic. The idea is quite simple: in any formula, whenever we bind an event to a variable z , we require that the binder operator explicitly state the consistency of z with the free variables appearing in the remaining part of the formula. Specifically, for any subformula of the kind $(x, \overline{y} < a\ z)\psi$, we could require the free variables of ψ to be a subset of $x \cup y \cup \{z\}$. In this way we are guaranteed that the event bound to z is either causally dependent or concurrent (hence consistent) with the events bound to the free variables of the formula. This essentially gives the same effect as restricting the semantics to legal pairs. Note that for any formula in \mathcal{L} we can construct an equivalent formula satisfying the above constraint. The key observation is that a quantified formula $(x, \overline{y} < a\ z)\psi$ can be transformed into $\bigvee_{x', y' \text{ s.t. } x' \oplus y' = fv(\psi) \setminus (x \cup y)} (x \cup x', \overline{y \cup y'} < a\ z)\psi$, where the fact that z must be consistent with any variable in $fv(\psi)$ has been made explicit by requiring any such variable, not already in $x \cup y$, to be either a cause or concurrent with z , in any possible way.

3.2 Negation and Dual operators

Strictly speaking, the negation operator of \mathcal{L} does not behave as classical negation. In fact, if we take an open formula φ and a denotation (C, η) which is not legal for φ , then neither $\mathcal{E}, C \models_{\eta} \varphi$ nor $\mathcal{E}, C \models_{\eta} \neg\varphi$. As a concrete example, take $\varphi = \langle x \rangle \langle y \rangle \top$. Then in the PES \mathcal{E}_1 of Fig. 1, if η binds x and y to the conflicting events labelled a and c , respectively, then (\emptyset, η) is not legal for φ and we have $\mathcal{E}_1, \emptyset \not\models_{\eta} \varphi$ and $\mathcal{E}_1, \emptyset \not\models_{\eta} \neg\varphi$.

However, since a pair (C, η) is legal for φ if and only if it is legal for $\neg\varphi$, the following result holds true.

Lemma 3 (negation). *Let φ be a formula in \mathcal{L} , let \mathcal{E} be a PES and let $(C, \eta) \in lp(\varphi)$. Then $\mathcal{E}, C \models_{\eta} \varphi$ iff $\mathcal{E}, C \not\models_{\eta} \neg\varphi$.*

In particular, when φ is closed, given any PES \mathcal{E} , since any pair (C, η) is legal for φ , the above lemma implies that $\mathcal{E} \models \neg\varphi$ iff $\mathcal{E} \not\models \varphi$.

Using negation we can define operators which are dual to those in the logic. As usual, $\varphi \vee \psi$ can be defined by the formula $\neg(\neg\varphi \wedge \neg\psi)$ and F (false) by $\neg\top$. Moreover, we write

$$\begin{aligned} \{x, \overline{y} < a\ z\} \varphi & \quad \text{for the formula} \quad \neg((x, \overline{y} < a\ z) \neg\varphi). \\ [z] \varphi & \quad \text{for the formula} \quad \neg(\langle z \rangle \neg\varphi) \end{aligned}$$

The dual of the binder has a universal flavour. In fact its semantics can be expressed as follows:

$$\begin{aligned} \llbracket \{x, \bar{y} < a z\} \varphi \rrbracket^{\mathcal{E}} = \{ (C, \eta) \mid & (C, \eta) \in lp(\{x, \bar{y} < a z\} \varphi) \text{ and} \\ & \forall e \in E[C] \text{ such that} \\ & \lambda(e) = a \wedge \eta(x) < e \wedge \eta(y) \parallel e \\ & \text{it holds } (C, \eta[z \mapsto e]) \in \llbracket \varphi \rrbracket^{\mathcal{E}} \} \end{aligned}$$

i.e., $\mathcal{E}, C \models_{\eta} \{x, \bar{y} < a z\} \varphi$ when for all a -labelled events e in the future of C , and consistent with the events already bound to $fv(\varphi)$, caused by $\eta(x)$ and concurrent with $\eta(y)$, binding e to z the formula φ holds.

The semantics of $[\cdot]$, instead, can be defined as:

$$\begin{aligned} \llbracket [z] \varphi \rrbracket^{\mathcal{E}} = \{ (C, \eta) \mid & (C, \eta) \in lp([z] \varphi) \text{ and} \\ & \text{if } C \xrightarrow{\eta(z)} C' \text{ then } (C', \eta) \in \llbracket \varphi \rrbracket^{\mathcal{E}} \} \end{aligned}$$

namely, $\mathcal{E}, C \models_{\eta} [z] \varphi$ if, either $\eta(z)$ is not executable from C or it is executable and in the reached configuration φ holds.

The logic \mathcal{L} could be alternatively defined in positive form by including the dual operators and omitting negation. The syntax of the resulting logic, denoted \mathcal{L}^+ , would be as follows:

$$\varphi ::= \mathbf{T} \mid \mathbf{F} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid (x, \bar{y} < a z) \varphi \mid \{x, \bar{y} < a z\} \varphi \mid \langle z \rangle \varphi \mid [z] \varphi$$

Negation is then encodable in \mathcal{L}^+ by duality. Hereafter we will freely use the dual operators.

3.3 Examples and notation

In this subsection we provide some more examples illustrating the expressiveness of the logic. We start by introducing some handy notation, which will improve the readability of the formulae.

Immediate execution. We will write

$$\langle x, \bar{y} < a z \rangle \varphi \quad \text{for the formula} \quad (x, \bar{y} < a z) \langle z \rangle \varphi$$

that chooses an event e that is not in the current configuration but that is enabled by it, and immediately executes e . We also introduce a notation for the dual of $\langle x, \bar{y} < a z \rangle \varphi$, denoted $\llbracket x, \bar{y} < a z \rrbracket \varphi$ and defined in the obvious way.

Steps. We introduce a notation also to predicate the existence, resp., the immediate execution, of concurrent events, specifying also their dependencies. We will write

$$\begin{aligned} ((x, \bar{y} < a z) \otimes (x', \bar{y}' < b z')) \varphi \quad & \text{for the formula} \quad (x, \bar{y} < a z)(x', \bar{y}', z' < b z') \varphi \\ (\llbracket x, \bar{y} < a z \rrbracket \otimes \llbracket x', \bar{y}' < b z' \rrbracket) \varphi \quad & \text{for the formula} \quad ((x, \bar{y} < a z) \otimes (x', \bar{y}' < b z')) \langle z \rangle \langle z' \rangle \varphi \end{aligned}$$

to declare the existence, resp., the immediate execution, of two concurrent events, labelled a and b , which are bound to z and z' , and then φ holds. In particular, the ability to perform a step consisting of two concurrent events labelled by a and b is simply expressed by the formula $(\langle a x \rangle \otimes \langle b y \rangle) \mathbf{T}$. Clearly, this notation can be generalised to the quantification and the immediate execution of any number of concurrent events.

An analogous notation will be used for the dual operators, i.e., we will write

$$(\{x, \bar{y} < a z\} \otimes \{x', \bar{y}' < b z'\}) \varphi \quad \text{and} \quad (\llbracket x, \bar{y} < a z \rrbracket \otimes \llbracket x', \bar{y}' < b z' \rrbracket) \varphi$$

to say that for any pair of concurrent events, resp., executing any two concurrent events, labelled a and b , which are bound to z and z' , then φ holds.

Example 1 (interleaving vs. true-concurrency). Consider the PESS \mathcal{E}_6 and \mathcal{E}_7 in Fig. 2. They are equated by interleaving equivalences and distinguished by any true-concurrent equivalence. The formula $\varphi_1 = \langle a x \rangle \langle \bar{x} < b y \rangle \mathbf{T} = (\langle a x \rangle \otimes \langle b y \rangle) \mathbf{T}$ is true only on \mathcal{E}_7 , while $\varphi_2 = \langle a x \rangle \langle x < b y \rangle \mathbf{T}$ is true only on \mathcal{E}_6 .

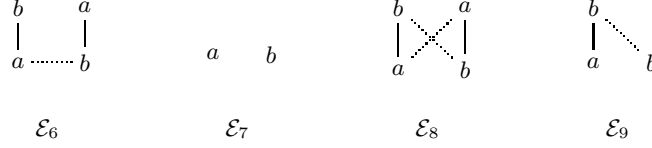


Fig. 2.

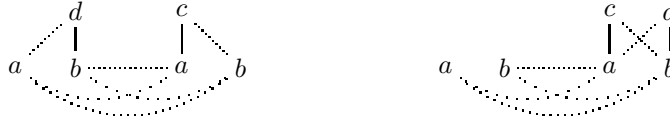
Wildcard operators. It is often useful to have a wildcard operator to refer to an event with any label. When the set of labels Λ is finite, we write

$$(x, \overline{y} < _z)\varphi$$

to denote the formula $\bigvee_{a \in \Lambda} (x, \overline{y} < a z)\varphi$, and similarly for the induced operators. For instance, the formula $(\langle _x \rangle \otimes \langle _y \rangle) \top \wedge \neg \langle _y_1 \rangle \otimes \langle _y_2 \rangle \otimes \langle _y_3 \rangle \top$ states that in the current state there is a step consisting of two concurrent events and this is the maximal size for a step. When the set of labels Λ is infinite the same wildcard operators are no longer expressible in the logic \mathcal{L} . However they can be added to \mathcal{L} while retaining all the results in the paper.

Example 2 (causality and concurrency). Consider the PESS \mathcal{E}_6 and \mathcal{E}_8 in Fig. 2. They are distinguished by any true-concurrent equivalence, but since they share the same causal structure, in order to pinpoint how they differ, the logic must be able to express the presence of two concurrent events. Logic \mathcal{L} can do this in a quite direct way, e.g., $\mathcal{E}_8 \models \langle a x \rangle \top \otimes \langle b y \rangle \top$, while $\mathcal{E}_6 \not\models \langle a x \rangle \top \otimes \langle b y \rangle \top$. On the other hand, PESS \mathcal{E}_7 and \mathcal{E}_9 , roughly speaking, exhibit the same concurrency and indeed they are equated by step bisimulation. However they have a different causal structure and thus they are distinguished by any equivalence which observes causality, e.g., pomset bisimilarity. The logic can take them apart by predicating directly about causality, e.g., \mathcal{E}_9 satisfies $\langle a x \rangle \langle x < b y \rangle \top$, while \mathcal{E}_7 does not.

Example 3 (conflicting futures). Consider the following two PESS, which can be proved to be hp-bisimilar but not hhp-bisimilar:



Intuitively, they differ since the causes of the c-labelled and d-labelled events are in conflict in the first PES and independent in the second one. This is captured by the formula $\varphi = ((a x) \otimes (b y))((x < c z_1) \top \wedge (y < d z_2) \top)$, which is satisfied only by the right-most PES. Notice that the formula φ exploits the ability of the logic \mathcal{L} of quantifying over events in conflict with previously bound events: formula φ is satisfied in the rightmost PES by binding x and y to the rightmost a-labelled and b-labelled events; then z_1 and z_2 are bound to events which are in conflict with either x or y . For this, the possibility of quantifying over an event without executing it is essential: the formula $\varphi' = (\langle a x \rangle \otimes \langle b y \rangle)((x < c z_1) \top \wedge (y < d z_2) \top)$ would be false for both PESS since the execution of the first two events leads to a configuration that is no further extensible.

As a last example, consider the two CCS processes $P = a|(b+c) + a|b + b|(a+c)$ and $Q = a|(b+c) + b|(a+c)$. They contain no causal dependencies, but they exhibit a different interplay between concurrency and branching. Accordingly, the corresponding PESS can be proved to be hp-bisimilar but not hhp-bisimilar. Intuitively, this difference comes from the fact that only the process P includes two concurrent events a and b such that, independently from their order of execution, no c-labelled event will be enabled. Such a difference can be expressed in \mathcal{L} by the formula $((a x) \otimes (b y))(\neg(\overline{x} < c z) \top \wedge \neg(\overline{y} < c z') \top)$, which is satisfied only by the PES corresponding to P .

4 A logical characterisation of hhp-bisimilarity

We next study the logical equivalence induced by \mathcal{L} . We have already argued that no formula in \mathcal{L} distinguishes the PESs a and $a\#a$, hence the logical equivalence induced by \mathcal{L} is surely coarser than isomorphism. In this section we will show that it coincides with hhp-bisimilarity.

Since later we will also identify suitable fragments of \mathcal{L} corresponding to coarser equivalences, we define logical equivalence for fragments of \mathcal{L} .

Definition 12 (logical equivalence). *Let \mathcal{L}' be a fragment of \mathcal{L} . We say that two PES $\mathcal{E}_1, \mathcal{E}_2$ are logically equivalent in \mathcal{L}' , written $\mathcal{E}_1 \equiv_{\mathcal{L}'} \mathcal{E}_2$ when they satisfy the same closed formulae.*

We first prove that two PES's satisfying the same formulae in \mathcal{L} are hhp-bisimilar.

Proposition 1. *Let \mathcal{E}_1 and \mathcal{E}_2 be PESs such that $\mathcal{E}_1 \equiv_{\mathcal{L}} \mathcal{E}_2$, then $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$.*

Proof. We first introduce some notation. Let us fix an injective environment $\eta_1 : E_1 \rightarrow Var$. Then given an event $e_1 \in E_1$, we write x_{e_1} to denote the only variable such that $\eta_1(x_{e_1}) = e_1$. Similarly, for a configuration $C_1 = \{e_1, \dots, e_n\}$ we denote by X_{C_1} the set of variables $\{x_{e_1}, \dots, x_{e_n}\}$. Observe that (\emptyset, η_1) is a legal pair for any formula $\varphi \in \mathcal{L}$ such that $fv(\varphi) \subseteq X_{C_1}$. Consider the posetal relation $R \subseteq \mathcal{C}(\mathcal{E}_1) \bar{\times} \mathcal{C}(\mathcal{E}_2)$ defined by:

$$R = \{ (C_1, f, C_2) \mid \forall \psi \in \mathcal{L}. fv(\psi) \subseteq X_{C_1} \quad (\mathcal{E}_1, \emptyset \models_{\eta_1} \psi \text{ iff } \mathcal{E}_2, \emptyset \models_{f \circ \eta_1} \psi) \} \quad (1)$$

where, given $f : C_1 \rightarrow C_2$ isomorphism of pomsets, by $f \circ \eta_1$ we denote an environment such that $f \circ \eta_1(x) = f(\eta_1(x))$ for $x \in X_{C_1}$ and $f \circ \eta_1(x)$ has any value, otherwise. Note that this does not introduce ambiguities, since, by Lemma 2, the semantics of φ only depends on the value of the environment on $fv(\varphi)$ and $fv(\varphi) \subseteq X_{C_1}$ by construction.

Observe that, since by hypothesis $\mathcal{E}_1 \equiv_{\mathcal{L}} \mathcal{E}_2$, we have that $(\emptyset, \emptyset, \emptyset) \in R$. Hence in order to conclude it is sufficient to show that R is a hhp-bisimulation.

– R is downward closed

Take $(C_1, f, C_2) \in R$ and consider $(C'_1, f', C'_2) \subseteq (C_1, f, C_2)$ pointwise. We have to show that $(C'_1, f', C'_2) \in R$.

Let ψ be any formula such that $fv(\psi) \subseteq X_{C'_1}$. Since $C'_1 \subseteq C_1$, clearly $fv(\psi) \subseteq X_{C_1}$ and thus, since $(C_1, f, C_2) \in R$, by definition of R (1), we have that

$$\mathcal{E}_1, \emptyset \models_{\eta_1} \psi \quad \text{iff} \quad \mathcal{E}_2, \emptyset \models_{f \circ \eta_1} \psi,$$

Moreover, since $fv(\psi) \subseteq X_{C'_1}$, $\eta_1(X_{C'_1}) = C'_1$ and $f' = f|_{C'_1}$, we have that $(f \circ \eta_1)|_{fv(\psi)} = (f' \circ \eta_1)|_{fv(\psi)}$ and thus by Lemma 2,

$$\mathcal{E}_2, \emptyset \models_{f \circ \eta_1} \psi \quad \text{iff} \quad \mathcal{E}_2, \emptyset \models_{f' \circ \eta_1} \psi$$

Summing up, for any ψ such that $fv(\psi) \subseteq X_{C'_1}$, it holds $\mathcal{E}_1, \emptyset \models_{\eta_1} \psi$ iff $\mathcal{E}_2, \emptyset \models_{f' \circ \eta_1} \psi$. Therefore $(C'_1, f', C'_2) \in R$, as desired.

– R is a hp-bisimulation

We have to show that given $(C_1, f, C_2) \in R$, if $C_1 \xrightarrow{e} C'_1$ then there exists a transition $C_2 \xrightarrow{g} C'_2$ such that $f' = f[e \mapsto g] : C'_1 \rightarrow C'_2$ is an isomorphism of pomsets (hence in particular $\lambda_1(e) = \lambda_2(g)$) and $(C'_1, f', C'_2) \in R$.

We proceed by contradiction. Since all PESs are assumed to be image finite, there are finitely many transitions $C_2 \xrightarrow{g^i} C_2^i$, $i \in \{1, \dots, n\}$, such that $C'_1 \sim C_2^i$ (as pomsets). By contradiction assume that $(C'_1, f^i, C_2^i) \notin R$ for any $i \in \{1, \dots, n\}$. Hence, by definition of R (1), there exists a formula ψ^i such that

$$\mathcal{E}_1, \emptyset \models_{\eta_1} \psi^i \quad \text{and} \quad \mathcal{E}_2, \emptyset \not\models_{f^i \circ \eta_1} \psi^i$$

where $fv(\psi^i) \subseteq X_{C'_1} = X_{C_1} \cup \{x_e\}$ and $f^i = f[e \mapsto g^i]$. Observe that it could either be that $\mathcal{E}_1, \emptyset \not\models_{\eta_1} \psi^i$ and $\mathcal{E}_2, \emptyset \models_{f^i \circ \eta_1} \psi^i$, but we can reduce to the case above by taking the negation of ψ^i . Then consider the formula

$$\varphi = (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} x_e) (\langle X_{C_1} \rangle \langle x_e \rangle \top \wedge \psi^1 \wedge \dots \wedge \psi^n)$$

where $\mathbf{a} = \lambda_1(e)$ and the $\mathbf{x}, \mathbf{y} \subseteq X_{C_1}$ are such that $\eta_1(\mathbf{x})$ is the set of causes of e in C_1 and $\eta_1(\mathbf{y})$ is the set of events in C_1 which are concurrent with e . Note that

$$fv(\varphi) = (X_{C_1 \cup e} \cup \bigcup_{i=1}^n fv(\psi_i)) \setminus \{x_e\} \cup \mathbf{x} \cup \mathbf{y} \subseteq X_{C_1}$$

Moreover, it is easy to see that $\mathcal{E}_1, \emptyset \models_{\eta_1} \varphi$ and $\mathcal{E}_2, \emptyset \not\models_{f \circ \eta_1} \varphi$, contradicting the hypotheses.

The fact that also the converse holds, i.e., if $C_2 \xrightarrow{g} C'_2$ then there exists a transition $C_1 \xrightarrow{e} C'_1$ such that $f' = f[e \mapsto g] : C'_1 \rightarrow C'_2$ is an isomorphism of pomsets and $(C'_1, f', C'_2) \in R$, can be proved analogously. \square

In order to prove the converse, i.e., the fact that hhp-bisimilar PESS satisfy the same \mathcal{L} formulae, we first adapt a lemma from [Bed91,vGG01] which will be useful in the sequel.

Lemma 4 (hhp-bisimilarity as a pes). *Let $\mathcal{E}_1, \mathcal{E}_2$ be PESS such that $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$ and let R be a hhp-bisimulation. Then there exists a PES $\mathcal{E}_R = \langle E_R, \leq_R, \#_R, \lambda_R \rangle$ such that for $i \in \{1, 2\}$*

- $\mathcal{E}_i \sim_{hhp} \mathcal{E}_R$
- *there are surjective maps $f_R^i : E_R \rightarrow E_i$ such that $\{(C, f_R^i|_C, f_R^i(C)) \mid C \in \mathcal{C}(\mathcal{E}_R)\}$ is a hhp-bisimulation.*

Additionally, each f_R^i preserves labels, \leq and \parallel , maps configurations to configurations and it is injective on pairwise consistent sets of events.

Proof (Sketch, from [Bed91,vGG01]). We just recall the definition of $\mathcal{E}_R = \langle E_R, \leq_R, \#_R, \lambda_R \rangle$:

- $E_R = \{(e_1, f, e_2) \mid ([e_1], f, [e_2]) \in R\}$,
- $(e_1, f, e_2) \leq_R (e'_1, f', e'_2)$ if $f \subseteq f'$,
- $f \#_R f'$ if there exists no $(C, g, D) \in R$ such that $([e_1], f, [e_2]), ([e'_1], f', [e'_2]) \subseteq (C, g, D)$ pointwise,
- $\lambda_R(e_1, f, e_2) = \lambda_1(e_1)$.

The maps $f_R^1 : E_R \rightarrow E_1$ and $f_R^2 : E_R \rightarrow E_2$ are just the projections on the first and third components, respectively. \square

Proposition 2. *Let \mathcal{E}_1 and \mathcal{E}_2 be PESS such that $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$. Then $\mathcal{E}_1 \equiv_{\mathcal{L}} \mathcal{E}_2$.*

Proof. Let R be a hhp-bisimulation relating \mathcal{E}_1 and \mathcal{E}_2 . By Lemma 4, it is not restrictive to assume that $R = \{(C_1, f|_{C_1}, f(C_1))\}$, where $f : E_1 \rightarrow E_2$ is a surjective map satisfying the conditions in the statement of the lemma. Then it is sufficient to prove that for any formula $\varphi \in \mathcal{L}$, for any $(C_1, \eta) \in lp(\varphi)$

$$\mathcal{E}_1, C_1 \models_{\eta} \varphi \quad \text{iff} \quad \mathcal{E}_2, f(C_1) \models_{f \circ \eta} \varphi$$

Indeed, since (\emptyset, \emptyset) is legal for any closed formula, it implies in particular that \mathcal{E}_1 and \mathcal{E}_2 satisfy the same closed formulae, i.e., $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$ as desired.

First of all note that (C_1, η) is a legal pair for φ iff $(f(C_1), f \circ \eta)$ is a legal pair for φ since f preserves consistency (as it preserves causality and concurrency).

We proceed by induction on the formula φ :

- $\varphi = \top$, $\varphi = \varphi_1 \wedge \varphi_2$, $\varphi = \neg \varphi_1$
Immediate.

– $\varphi = (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} \ z) \psi$

Assume that $\mathcal{E}_1, C_1 \models_{\eta} \varphi$. Hence, by definition of the semantics, there exists an event $e \in E_1[C_1]$, such that $\lambda_1(e) = \mathbf{a}$, $\eta(\mathbf{x}) \leq e$, $\eta(\mathbf{y}) \parallel e$ and

$$\mathcal{E}_1, C_1 \models_{\eta'} \psi \quad (2)$$

where $\eta' = \eta[z \mapsto e]$.

By (2) and Lemma 1, (C_1, η') is legal for ψ . Hence by inductive hypothesis $\mathcal{E}_2, f(C_1) \models_{f \circ \eta'} \psi$, with $f \circ \eta' = (f \circ \eta)[z \mapsto f(e)]$.

Since, by Lemma 4, f is injective on pairwise consistent sets of events, $f(e) \in E_2[f(C_1)]$. Additionally, again by Lemma 4, since f preserves labels, \leq and \parallel we have that $\lambda_2(f(e)) = \lambda_1(e) = \mathbf{a}$ and $f(\eta(\mathbf{x})) \leq f(e)$, $f(\eta(\mathbf{y})) \parallel f(e)$. Therefore we conclude that, as desired

$$\mathcal{E}_2, f(C_1) \models_{f \circ \eta} \varphi.$$

Vice versa, let $\mathcal{E}_2, f(C_1) \models_{f \circ \eta} \varphi$. Therefore there exists an event $g \in E_2[f(C_1)]$, such that $\lambda_2(g) = \mathbf{a}$, $f(\eta(\mathbf{x})) \leq g$ and $f(\eta(\mathbf{y})) \parallel g$ and $\mathcal{E}_2, f(C_1) \models_{\eta_2} \psi$, where $\eta_2 = (f \circ \eta)[z \mapsto g]$.

From the fact that $\mathcal{E}_2, f(C_1) \models_{f \circ \eta} \varphi$ and Lemma 1, we have that $(f(C_1), f \circ \eta)$ is legal for φ . This implies, in particular, that that $D_2 = f(C_1) \cup [f(\eta(\mathbf{x} \cup \mathbf{y}))]$ is a configuration. Notice that also $D_2 \cap g$, hence if we let $X_2 = [g] \setminus D_2$ we have

$$D_2 \xrightarrow{X_2} D'_2 \quad (3)$$

Now, since by hypothesis (C_1, η) is legal for φ , and since $\mathbf{x} \cup \mathbf{y} \subseteq fv(\varphi)$, we know that $C_1 \cup \eta(\mathbf{x} \cup \mathbf{y})$ is pairwise consistent. It follows that $D_1 = C_1 \cup [\eta(\mathbf{x} \cup \mathbf{y})]$ is a configuration. Since, by Lemma 4, f is injective on pairwise consistent sets and preserves causality, $[f(\eta(\mathbf{x} \cup \mathbf{y}))] = f([\eta(\mathbf{x} \cup \mathbf{y})])$ and thus $D_2 = f(D_1)$, which means that $(D_1, f|_{D_1}, f(D_1)) \in R$. Therefore, since R is a hhp-bisimulation, there is a pomset transition simulating (3):

$$D_1 \xrightarrow{X_1} D'_1 \quad (4)$$

such that $(D'_1, f|_{D'_1}, D'_2) \in R$. In particular, since $f|_{D'_1}$ is an isomorphism of pomsets between D'_1 and D'_2 , if we take the (unique) $e \in X_1$ such that $f(e) = g$, it holds that $\lambda_1(e) = \lambda_2(g) = \mathbf{a}$, $\eta(\mathbf{x}) \leq e$ and $\eta(\mathbf{y}) \parallel e$. Now, if we let $\eta' = \eta[z \mapsto e]$, since $f \circ \eta' = \eta_2$ and $(C_1, \eta') \in lp(\varphi)$ by construction, we can use the inductive hypothesis, together with $\mathcal{E}_2, f(C_1) \models_{\eta_2} \psi$, to conclude $\mathcal{E}_1, C_1 \models_{\eta'} \psi$. Hence

$$\mathcal{E}_1, C_1 \models_{\eta} \varphi$$

– $\varphi = \langle \mathbf{x} \rangle \psi$

Assume that $\mathcal{E}_1, C_1 \models_{\eta} \varphi$, where $(C_1, \eta) \in lp(\varphi)$. By definition of the semantics this means that

$$C_1 \xrightarrow{\eta(\mathbf{x})} C'_1$$

and $\mathcal{E}_1, C'_1 \models_{\eta} \psi$.

Since R is a hhp-bisimulation, we have that

$$f(C_1) \xrightarrow{f(\eta(\mathbf{x}))} f(C'_1)$$

and, by inductive hypothesis, $\mathcal{E}_2, f(C'_1) \models_{f \circ \eta} \psi$. This implies that, as desired

$$\mathcal{E}_2, f(C_1) \models_{f \circ \eta} \varphi$$

Vice versa, let $\mathcal{E}_2, f(C_1) \models_{f \circ \eta} \varphi$, where (C_1, η) is legal for φ . By definition of the semantics this means that

$$f(C_1) \xrightarrow{f(\eta(x))} C'_2$$

and $\mathcal{E}_2, C'_2 \models_{f \circ \eta} \psi$.

Since (C_1, η) is legal for φ , we know that $\eta(x)$ is consistent with C_1 . Moreover, $C_1 \cup \{\eta(x)\}$ is causally closed, otherwise, since f preserves causality and it is injective on pairwise consistent sets, also $f(C_1 \cup \eta(x)) = C_2 \cup f(\eta(x)) = C'_2$ would not be causally closed.

Hence $C'_1 = C_1 \cup \{\eta(x)\}$ is a configuration and thus

$$C_1 \xrightarrow{\eta(x)} C'_1$$

and clearly $f(C'_1) = C'_2$. Moreover, by inductive hypothesis, $\mathcal{E}_1, C'_1 \models_{\eta} \psi$. Hence, as desired

$$\mathcal{E}_1, C_1 \models_{\eta} \varphi$$

□

Propositions 2 and 1 together say that hhp-bisimilarity is the logical equivalence of \mathcal{L} .

Theorem 1 (hhp-bisimilarity). *Let \mathcal{E}_1 and \mathcal{E}_2 be PESS. Then $\mathcal{E}_1 \sim_{hhp} \mathcal{E}_2$ iff $\mathcal{E}_1 \equiv_{\mathcal{L}} \mathcal{E}_2$.*

5 From Hennessy-Milner logic to HP-logic

Hhp-bisimilarity is the finest equivalence in the spectrum of true concurrent equivalences proposed in [vGG01]. Interestingly enough, coarser equivalences such as step, pomset and hp-bisimilarity, can be captured by suitable fragments of \mathcal{L} summarised in Fig. 3, which can be viewed as the logical counterpart of the true concurrent spectrum.

Note that, in each of these fragments after predicating the existence of an event we must execute it. As a consequence, differently from what happens in the full logic, in the fragments it is impossible to refer to events in conflict with already observed events. Intuitively, this says that behavioural equivalences up to hp-bisimilarity observe events only by executing them. Hence they cannot fully capture the interplay between concurrency and branching, which is indeed distinctive of hhp-equivalence.

HM Logic	\mathcal{L}_{HM}	$\varphi ::= \langle \mathbf{a} x \rangle \varphi \mid \varphi \wedge \varphi \mid \neg \varphi \mid \mathbf{T}$
Step Logic	\mathcal{L}_s	$\varphi ::= (\langle \mathbf{a}_1 x_1 \rangle \otimes \cdots \otimes \langle \mathbf{a}_n x_n \rangle) \varphi \mid \varphi \wedge \varphi \mid \neg \varphi \mid \mathbf{T}$
Pomset Logic	\mathcal{L}_p	$\varphi ::= \langle \mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z \rangle \varphi \mid \neg \varphi \mid \varphi \wedge \varphi \mid \mathbf{T}$ where \neg, \wedge are used only on closed formulae.
HP Logic	\mathcal{L}_{hp}	$\varphi ::= \langle \mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z \rangle \varphi \mid \neg \varphi \mid \varphi \wedge \varphi \mid \mathbf{T}$

Fig. 3. Fragments of \mathcal{L} corresponding to various behavioural equivalences

5.1 Hennessy-Milner Logic

A first simple observation is that standard Hennessy-Milner logic can be recovered as the fragment of \mathcal{L} where only the derived modality $\langle \mathbf{a} x \rangle \varphi$ (with no references to causally dependent/concurrent events) is allowed. In words, whenever we state the existence of an enabled event we are forced to execute it. Moreover, since no dependencies can be expressed, the bound variable x is irrelevant. The induced logical equivalence is thus bisimilarity [HM85] (recall that we consider only image finite PES's).

5.2 Step logic

A fragment \mathcal{L}_s corresponding to step bisimilarity naturally arises as a generalisation of HM logic where we can refer to sets of concurrently enabled events. More precisely, as shown in Fig. 3, \mathcal{L}_s is the fragment of \mathcal{L} where only the derived modality $\langle\!\langle \mathbf{a}_1 x_1 \rangle\!\rangle \otimes \cdots \otimes \langle\!\langle \mathbf{a}_n x_n \rangle\!\rangle$ is used, allowing to predicate on the possibility of performing a parallel step, but without any reference to causal dependencies. Note that all formulae in \mathcal{L}_s are closed, and thus environments are irrelevant in their semantics (as well as the names of the bound variables). Given a PES \mathcal{E} , a configuration C and a formula φ we will write $\mathcal{E}, C \models \varphi$ when $\mathcal{E}, C \models_\eta \varphi$ for some η .

As an example, consider the two PESs \mathcal{E}_6 and \mathcal{E}_7 in Fig. 2. They are bisimilar but not step bisimilar since only \mathcal{E}_7 can execute the step consisting of \mathbf{a} and \mathbf{b} ; accordingly, the formula $\langle\!\langle \mathbf{a} \rangle\!\rangle \otimes \langle\!\langle \mathbf{b} \rangle\!\rangle$ in \mathcal{L}_s is true only on \mathcal{E}_7 .

Lemma 5. *Let \mathcal{E}_1 and \mathcal{E}_2 be PESs and let $C_i \in \mathcal{C}(\mathcal{E}_i)$, $i \in \{1, 2\}$ be configurations. Then there exists a step bisimulation R such that $(C_1, C_2) \in R$ iff for any $\varphi \in \mathcal{L}_s$, $\mathcal{E}_1, C_1 \models \varphi \Leftrightarrow \mathcal{E}_2, C_2 \models \varphi$.*

Proof. (\Rightarrow) Assume that $(C_1, C_2) \in R$ for some step bisimulation R . The proof that for all $\varphi \in \mathcal{L}_s$, $\mathcal{E}_1, C_1 \models \varphi$ iff $\mathcal{E}_2, C_2 \models \varphi$ can be carried out by induction on the structure of φ .

We only show the non-trivial case where $\varphi = (\langle\!\langle \mathbf{a}_1 x_1 \rangle\!\rangle \otimes \cdots \otimes \langle\!\langle \mathbf{a}_n x_n \rangle\!\rangle) \psi$. Assume that $\mathcal{E}_1, C_1 \models \varphi$. Hence there is a step $C_1 \xrightarrow{\{e_1, \dots, e_n\}} C'_1$ where $\lambda_1(e_i) = \mathbf{a}_i$ for $i \in \{1, \dots, n\}$ and

$$\mathcal{E}_1, C'_1 \models \psi. \quad (5)$$

Since $(C_1, C_2) \in R$, also C_2 can perform an analogous step

$$C_2 \xrightarrow{\{e'_1, \dots, e'_n\}} C'_2$$

with $\lambda_2(e'_i) = \mathbf{a}_i$ for $i \in \{1, \dots, n\}$ and $(C'_1, C'_2) \in R$. Additionally, by (5), using the induction hypothesis, we have that $\mathcal{E}_2, C'_2 \models \psi$. Therefore we conclude $\mathcal{E}_2, C_2 \models \varphi$.

(\Leftarrow) We prove that the relation

$$R = \{(C_1, C_2) \mid \forall \varphi \in \mathcal{L}_s \ (\mathcal{E}_1, C_1 \models \varphi \text{ iff } \mathcal{E}_2, C_2 \models \varphi)\}$$

is a step bisimulation.

We proceed by contradiction. Let $(C_1, C_2) \in R$, let $C_1 \xrightarrow{X} C'_1$, where X is a step, and assume that for all Y such that $C_2 \xrightarrow{Y} C'_2$ and $X \sim Y$ as pomsets it does not hold that $(C'_1, C'_2) \in R$. Hence there exists a formula ψ such that $\mathcal{E}_1, C'_1 \models \psi$ and $\mathcal{E}_2, C'_2 \not\models \psi$.

Since our PESs are assumed to be image finite, the number of possible steps $C_2 \xrightarrow{Y} C'_2$, with $X \sim Y$ is finite. Let $C_2 \xrightarrow{Y^i} C_2^i$, for $i \in \{1, \dots, k\}$, be such steps and let ψ^i be the formulae such that $\mathcal{E}_1, C'_1 \models \psi^i$ and $\mathcal{E}_2, C_2^i \not\models \psi^i$. If we define

$$\psi = (\langle\!\langle \mathbf{a}_1 x_1 \rangle\!\rangle \otimes \cdots \otimes \langle\!\langle \mathbf{a}_n x_n \rangle\!\rangle) (\psi^1 \wedge \dots \wedge \psi^k)$$

we have that $\mathcal{E}_1, C_1 \models \psi$ while $\mathcal{E}_2, C_2 \not\models \psi$, which gives the desired contradiction. \square

Now it is immediate to conclude that the following holds.

Theorem 2 (step bisimilarity). *Let \mathcal{E}_1 and \mathcal{E}_2 be PESs. Then $\mathcal{E}_1 \sim_s \mathcal{E}_2$ iff $\mathcal{E}_1 \equiv_{\mathcal{L}_s} \mathcal{E}_2$.*

5.3 Pomset logic

The logic \mathcal{L}_p for pomset bisimilarity in Fig. 3 consists of a fragment of \mathcal{L} where, still an event must be immediately executed when quantified, but it is possible to refer to dependencies between events. However, propositional connectives (negation and conjunction) can be used only on closed formulae.

Roughly speaking, in \mathcal{L}_p closed subformulae characterise the execution of a pomset. Then, the requirement that the propositional operators are used only on closed subformulae prevents pomset transitions from being causally linked to the events in the past. These ideas are formalised by the results below.

First observe that a closed formula in \mathcal{L}_p has always the shape

$$\langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle \psi$$

where, if we let $Z = \{z_1, \dots, z_n\}$, then $\mathbf{x}_i, \mathbf{y}_i \subseteq Z$ for any $i \in \{1, \dots, n\}$. We next prove that the prefix $\langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle$ intuitively corresponds to the execution of a class of pomsets (not a single one, since the relation between some events might be not specified). More precisely, in the situation above let $Pom(\langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle)$ denote the class of pomsets (Z, \leq, λ) such that $Z = \{z_1, \dots, z_n\}$ and for $i \in \{1, \dots, n\}$, $\lambda(z_i) = \mathbf{a}_i$ and given any $z \in Z$

- $z \in \mathbf{x}_i$ implies $z \leq z_i$,
- $z \in \mathbf{y}_i$ implies $z \not\leq z_i$.

With this definition it is immediate to show that the following result holds.

Lemma 6. *Let $\varphi = \langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle \psi$ be a closed formula in \mathcal{L}_p . Then*

$$\begin{aligned} \mathcal{E}, C \models_{\eta} \varphi \quad \text{iff} \quad & C \xrightarrow{X} C' \text{ where } X = \{e_1, \dots, e_n\} \text{ is a pomset s.t. } X \sim (Z, \leq, \lambda) \\ & \text{for some } (Z, \leq, \lambda) \in Pom(\langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle) \\ & \text{and } \mathcal{E}, C' \models_{\eta'} \psi, \text{ with } \eta' = \eta[z_1 \mapsto e_1, \dots, z_n \mapsto e_n] \end{aligned}$$

Proof. Routine induction on n . □

Next we observe that, in particular, the execution of a single pomset can be exactly characterised by a corresponding formula in \mathcal{L}_p .

Definition 13 (pomsets as formulae in \mathcal{L}_p). *Let $p_x = (\{x_1, \dots, x_n\}, \leq_{p_x}, \lambda_{p_x})$ be a labelled poset, whose elements $\{x_1, \dots, x_n\} \subseteq \text{Var}$ are variables ordered by \leq_{p_x} . Given a formula $\varphi \in \mathcal{L}_p$, we denote by $\langle p_x \rangle \varphi$ the formula inductively defined as follows. If p_x is empty then $\langle p_x \rangle \varphi = \varphi$. If $p_x = p'_x \cup \{x\}$, where x is maximal with respect to \leq_{p_x} , if we let $\mathbf{y} = \{x' \in p'_x \mid x' \leq_{p_x} x\}$, $\mathbf{z} = p'_x \setminus \mathbf{y}$, and $\lambda_{p_x}(x) = \mathbf{a}$, then $\langle p_x \rangle \varphi = \langle p'_x \rangle \langle \mathbf{y}, \overline{\mathbf{z}} < \mathbf{a} x \rangle \varphi$.*

The fact that pomset formulae as defined above have exactly the intended semantics immediately follows from Lemma 6.

Lemma 7 (pomsets in \mathcal{L}_p). *Let \mathcal{E} be a PES and let $C \in \mathcal{C}(\mathcal{E})$ be a configuration. Given a labelled poset $p_x = (\{x_1, \dots, x_n\}, \leq_{p_x}, \lambda_{p_x})$, then*

$$\begin{aligned} \mathcal{E}, C \models_{\eta} \langle p_x \rangle \varphi \quad \text{iff} \quad & C \xrightarrow{X} C' \text{ where } X = \{e_1, \dots, e_n\} \text{ is a pomset s.t. } X \sim p_x \\ & \text{and } \mathcal{E}, C' \models_{\eta'} \varphi, \text{ with } \eta' = \eta[x_1 \mapsto e_1, \dots, x_n \mapsto e_n] \end{aligned}$$

Proof. Just observe that $Pom(\langle p_x \rangle) = \{p_x\}$. Then the result is an instance of Lemma 6. □

Lemma 8. *Let \mathcal{E}_1 and \mathcal{E}_2 be PESs and let $C_i \in \mathcal{C}(\mathcal{E}_i)$, $i \in \{1, 2\}$, be configurations. Then there exists a pomset bisimulation R such that $(C_1, C_2) \in R$ iff for any $\varphi \in \mathcal{L}_p$, φ closed, $\mathcal{E}_1, C_1 \models_{\emptyset} \varphi \Leftrightarrow \mathcal{E}_2, C_2 \models_{\emptyset} \varphi$.*

Proof. (\Rightarrow) Let us prove that if $(C_1, C_2) \in R$, then for all closed formulae $\varphi \in \mathcal{L}_p$, we have that $\mathcal{E}_1, C_1 \models_{\emptyset} \varphi$ iff $\mathcal{E}_2, C_2 \models_{\emptyset} \varphi$.

The proof proceeds by induction on the structure of the formula φ . The cases in which φ is a conjunction, negation or true are trivial. In the remaining cases φ is a closed formula of the shape

$$\langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle \psi. \quad (6)$$

where ψ is closed.

Assume that $\mathcal{E}_1, C_1 \models_{\emptyset} \varphi$. Then, by Lemma 6, $C_1 \xrightarrow{X} C'_1$ where $X \sim (Z, \leq, \lambda)$ for some pomset $(Z, \leq, \lambda) \in \text{Pom}(\langle \mathbf{x}_1, \overline{\mathbf{y}}_1 < \mathbf{a}_1 z_1 \rangle \dots \langle \mathbf{x}_n, \overline{\mathbf{y}}_n < \mathbf{a}_n z_n \rangle)$. Additionally $\mathcal{E}_1, C'_1 \models_{\emptyset[z_1 \mapsto e_1, \dots, z_n \mapsto e_n]} \psi$, and thus $\mathcal{E}_1, C'_1 \models_{\emptyset} \psi$, by Lemma 2, since ψ is closed.

Since $(C_1, C_2) \in R$ and R is a pomset bisimulation, there is a pomset $Y = \{g_1, \dots, g_n\}$, isomorphic to X , and thus to (Z, \leq, λ) such that

$$C_2 \xrightarrow{Y} C'_2 \quad (7)$$

and $(C'_1, C'_2) \in R$. By inductive hypothesis, $\mathcal{E}_2, C'_2 \models_{\emptyset} \psi$. Again, since ψ is closed, by Lemma 2 it also holds $\mathcal{E}_2, C'_2 \models_{\emptyset[z_1 \mapsto g_1, \dots, z_n \mapsto g_n]} \psi$. This fact, together with (7), allows us to conclude, by Lemma 6, that $\mathcal{E}_2, C_2 \models_{\emptyset} \varphi$, as desired.

(\Leftarrow) The proof is analogous to that of Lemma 5, i.e., we show that the relation

$$R = \{(C_1, C_2) \mid \forall \varphi \in \mathcal{L}_p, \varphi \text{ closed, } \mathcal{E}_1, C_1 \models_{\emptyset} \varphi \text{ iff } \mathcal{E}_2, C_2 \models_{\emptyset} \varphi\}$$

is a pomset bisimulation.

We proceed by contradiction. Let $(C_1, C_2) \in R$, let $C_1 \xrightarrow{X} C'_1$, where X is a pomset, and assume that for all Y such that $C_2 \xrightarrow{Y} C'_2$ and $X \sim Y$ there exists a closed formula ψ such that $\mathcal{E}_1, C'_1 \models \psi$ and $\mathcal{E}_2, C'_2 \not\models \psi$.

Since our PESS are assumed to be image finite, there are finitely many such pomset transitions $C_2 \xrightarrow{Y^i} C'_2$, for $i \in \{1, \dots, k\}$. Let ψ^i be the formulae such that $\mathcal{E}_1, C'_1 \models \psi^i$ and $\mathcal{E}_2, C'_2 \not\models \psi^i$. If p_x is a labelled pomset of variables, such that $p_x \sim X$, let us define:

$$\psi = \langle p_x \rangle (\psi^1 \wedge \dots \wedge \psi^k)$$

Then by Lemma 7, we have that $\mathcal{E}_1, C_1 \models \psi$ while $\mathcal{E}_2, C_2 \not\models \psi$, which gives the desired contradiction. \square

The desired result now immediately follows.

Theorem 3 (pomset bisimilarity). *Let \mathcal{E}_1 and \mathcal{E}_2 be PESS. Then $\mathcal{E}_1 \sim_p \mathcal{E}_2$ iff $\mathcal{E}_1 \equiv_{\mathcal{L}_p} \mathcal{E}_2$.*

As an example, consider the two PESS \mathcal{E}_7 and \mathcal{E}_9 in Fig. 2. They are step bisimilar but not pomset bisimilar since only the second one can execute the pomset $p_{a < b} = (\{\mathbf{a}, \mathbf{b}\}, \mathbf{a} < \mathbf{b})$. Accordingly, the formula $\varphi = \langle p_{a < b} \rangle \top = \langle \mathbf{a} x \rangle \langle x < \mathbf{b} y \rangle \top$ in \mathcal{L}_p , is satisfied only by \mathcal{E}_9 .

5.4 History preserving logic

The fragment \mathcal{L}_{hp} corresponding to hp-bisimilarity is essentially the same as for pomset logic, where we relax the condition asking that the propositional connectives are applied only to closed formulae. Intuitively, in this way a formula $\varphi \in \mathcal{L}_{hp}$, besides expressing the possibility of executing a pomset p_x , also predicates about its dependencies with previously executed events (bound to the free variables of φ).

The following two PESS can be proved to be pomset equivalent but not hp-equivalent:



Intuitively, they allow the same pomset transitions, but they have a different “causal branching”. Indeed, only in the left-most PESS after the execution of an \mathbf{a} -labelled event we can choose between an independent and a dependent \mathbf{b} -labelled event. In the rightmost PES the choice is already taken with the execution of \mathbf{a} . Formally, the formula $\langle \mathbf{a} x \rangle (\langle \overline{\mathbf{a}} < \mathbf{b} y \rangle \wedge (\langle x < \mathbf{b} z \rangle)$ in \mathcal{L}_{hp} is true only on the left-most PES.

We start with the following lemma, whose proof is immediate, that makes explicit the semantics of the induced operator $\langle \mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z \rangle$.

Lemma 9 (events with their history in the logic). *Given a PES \mathcal{E} , $C \in \mathcal{C}(\mathcal{E})$, $\varphi \in \mathcal{L}_{hp}$ and (C, η) a legal pair for φ :*

$$\mathcal{E}, C \models_{\eta} \langle \mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z \rangle \varphi \quad \text{iff} \quad \text{there is an event } e \text{ such that } C \xrightarrow{e} C', \lambda(e) = \mathbf{a}, \\ \eta(\mathbf{x}) \leq e, \eta(\mathbf{y}) \parallel e \text{ and } C' \models_{\eta'} \varphi, \text{ where } \eta' = \eta[z \mapsto e].$$

Lemma 10. *Let \mathcal{E}_1 and \mathcal{E}_2 be PESS, let $C_i \in \mathcal{C}(\mathcal{E}_i)$, for $i \in \{1, 2\}$, be configurations, and let $f : C_1 \rightarrow C_2$ be an isomorphism of pomsets. Then the following are equivalent:*

1. *there is a hp-bisimulation R such that $(C_1, f, C_2) \in R$*
2. *for any $\varphi \in \mathcal{L}_{hp}$ and η such that $\eta(fv(\varphi)) \subseteq C_1$, it holds $\mathcal{E}_1, C_1 \models_{\eta} \varphi \Leftrightarrow \mathcal{E}_2, C_2 \models_{f \circ \eta} \varphi$.*

Proof. $(1 \Rightarrow 2)$ Let R be a hp-bisimulation and take $(C_1, f, C_2) \in R$. Let us show that for any $\varphi \in \mathcal{L}_{hp}$ and (C_1, η) such that $\eta(fv(\varphi)) \subseteq C_1$.

$$\mathcal{E}_1, C_1 \models_{\eta} \varphi \quad \text{iff} \quad \mathcal{E}_2, C_2 \models_{f \circ \eta} \varphi.$$

We proceed by induction on the structure of the formula φ . We focus on the only non-trivial case where $\varphi = \langle \mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z \rangle \psi$. If $\mathcal{E}_1, C_1 \models_{\eta} \varphi$, then by Lemma 9 there is an event $e \in E_1[C_1]$ such that

$$C_1 \xrightarrow{e} C'_1 \tag{8}$$

with $\lambda_1(e) = \mathbf{a}$, $\eta(\mathbf{x}) \leq e$, $\eta(\mathbf{y}) \parallel e$ and $\mathcal{E}_1, C'_1 \models_{\eta'} \psi$ where $\eta' = \eta[z \mapsto e]$.

Since $(C_1, f, C_2) \in R$, there exists an event $g \in E_2$ such that

$$C_2 \xrightarrow{g} C'_2 \tag{9}$$

and $(C'_1, f', C'_2) \in R$, with $f' = f[e \mapsto g]$. Clearly, $g \in E_2[C_2]$ and, since f' is an isomorphism of configurations, we have that $\lambda_2(g) = \mathbf{a}$, $f(\eta(\mathbf{x})) \leq g$ and $f(\eta(\mathbf{y})) \parallel g$.

Note that $\eta'(fv(\psi)) \subseteq \eta'(fv(\varphi) \cup \{z\}) = \eta(fv(\varphi)) \cup \{e\} \subseteq C_1 \cup \{e\} = C'_1$. Thus, we can use the induction hypothesis to deduce that $\mathcal{E}_2, C'_2 \models_{f' \circ \eta'} \psi$. Therefore, by Lemma 9, we can conclude $\mathcal{E}_2, C_2 \models_{f \circ \eta} \varphi$.

The proof that $\mathcal{E}_2, C_2 \models_{f \circ \eta} \varphi$ implies $\mathcal{E}_1, C_1 \models_{\eta} \varphi$ is analogous and thus omitted.

$(1 \Leftarrow 2)$ As in Proposition 1 we fix an injective environment $\eta_1 : \text{Var} \rightarrow E_1$. Moreover, given an event $e_1 \in E_1$, we write x_{e_1} to denote the only variable such that $\eta_1(x_{e_1}) = e_1$. Similarly, for a configuration $C_1 = \{e_1, \dots, e_n\}$ we denote by X_{C_1} the set of variables $\{x_{e_1}, \dots, x_{e_n}\}$. Observe that (C_1, η_1) is a legal pair for any formula $\varphi \in \mathcal{L}$ such that $fv(\varphi) \subseteq X_{C_1}$.

In order to conclude, we show that the posetal relation

$$R = \{(C_1, f, C_2) \mid \forall \varphi \in \mathcal{L}_{hp}. fv(\varphi) \subseteq X_{C_1} \quad \mathcal{E}_1, C_1 \models_{\eta_1} \varphi \text{ iff } \mathcal{E}_2, C_2 \models_{f \circ \eta_1} \varphi\}$$

is a hp-bisimulation. Note that as in Proposition 1 with a slight abuse of notation, we denote by $f \circ \eta_1$ any environment η_2 such that $\eta_2(x) = f(\eta_1(x))$ for $x \in X_{C_1}$ and $\eta_2(x)$ has any value, otherwise.

We proceed by contradiction: assume that $(C_1, f, C_2) \in R$, $C_1 \xrightarrow{e_1} C'_1$ and for all e_2 such that $C_2 \xrightarrow{e_2} C'_2$ with $C'_1 \sim C'_2$ as pomsets, we have $(C'_1, f[e_1 \mapsto e_2], C'_2) \notin R$, i.e., there exists a formula ψ , with $fv(\psi) \subseteq X_{C'_1}$, such that $\mathcal{E}_1, C'_1 \models_{\eta_1} \psi$ and $\mathcal{E}_2, C_2 \not\models_{f \circ \eta_1} \psi$.

Since all PESS are assumed to be image finite, there are finitely many transitions

$$C_2 \xrightarrow{e_2^i} C_2^i, \quad i \in \{1, \dots, k\}$$

such that $f^i = f[e_1 \mapsto e_2^i] : C_1' \rightarrow C_2^i$ is an isomorphism of pomsets. Let ψ^i , for $i \in \{1, \dots, k\}$ be the formulae such that

$$\mathcal{E}_1, C_1' \models_{\eta_1} \psi^i \quad \text{and} \quad \mathcal{E}_2, C_2^i \not\models_{f^i \circ \eta_1} \psi^i$$

where $fv(\psi^i) \subseteq X_{C_1'} = X_{C_1} \cup \{x_{e_1}\}$. Now consider the formula

$$\varphi = \langle \mathbf{x}, \mathbf{y} < \mathbf{a} x_{e_1} \rangle (\psi^1 \wedge \dots \wedge \psi^k)$$

where $\mathbf{a} = \lambda_1(e_1)$ and the $\mathbf{x}, \mathbf{y} \subseteq X_{C_1}$ are such that $\eta_1(\mathbf{x})$ is the set of causes of e_1 in C_1 and $\eta_1(\mathbf{y})$ is the set of events in C_1 which are concurrent with e_1 . Note that $fv(\varphi) = (\bigcup_{i=1}^k fv(\psi^i)) \setminus \{x_{e_1}\} \cup \mathbf{x} \cup \mathbf{y} \subseteq X_{C_1}$

Then by Lemma 9 we have that $\mathcal{E}_1, C_1 \models_{\eta_1} \varphi$ and $\mathcal{E}_2, C_2 \not\models_{f \circ \eta_1} \varphi$, which gives the desired contradiction. \square

It is worth observing that the hp-bisimulation built in the previous proof relates two configurations C_1 and C_2 when they satisfy the same formulae, whereas the hhp-bisimulation built in the proof of Theorem 1 relates C_1 and C_2 when the same formulae are satisfied by the empty configuration (in an environment that binds free variables to C_1 , resp. C_2). Intuitively, this corresponds to the fact that for hp-bisimilarity one has to check only the future of a configuration, while for hhp-bisimilarity also alternative evolutions (hence evolutions from the past) of a configuration must be considered.

Theorem 4 (hp-bisimilarity). *Let \mathcal{E}_1 and \mathcal{E}_2 be PESSs. Then $\mathcal{E}_1 \sim_{hp} \mathcal{E}_2$ iff $\mathcal{E}_1 \equiv_{\mathcal{L}_{hp}} \mathcal{E}_2$.*

Proof. (\Rightarrow) Let $\mathcal{E}_1 \sim_{hp} \mathcal{E}_2$. Then there is an hp-bisimulation R such that $(\emptyset, \emptyset, \emptyset) \in R$. For all $\varphi \in \mathcal{L}_{hp}$, if φ is closed, i.e., $fv(\varphi) = \emptyset$, as an instance of Lemma 10, we obtained $\mathcal{E}_1, \emptyset \models_{\emptyset} \varphi$ iff $\mathcal{E}_2, \emptyset \models_{\emptyset} \varphi$. This amounts to $\mathcal{E}_1 \models \varphi$ iff $\mathcal{E}_2 \models \varphi$, i.e., $\mathcal{E}_1 \equiv_{\mathcal{L}_{hp}} \mathcal{E}_2$, as desired.

(\Leftarrow) Let $\mathcal{E}_1 \equiv_{\mathcal{L}_{hp}} \mathcal{E}_2$. Then, for any $\varphi \in \mathcal{L}_{hp}$ closed $\mathcal{E}_1 \models \varphi$ iff $\mathcal{E}_2 \models \varphi$. Since φ is closed, satisfaction does not depend on the environment, $\mathcal{E}_1, \emptyset \models_{\eta} \varphi$ iff $\mathcal{E}_2, \emptyset \models_{\eta} \varphi$ hence for any $\eta \in Env_{\mathcal{E}_1}$. Therefore, we can apply Lemma 10 to conclude that there exists a hp-bisimulation R such that $(\emptyset, \emptyset, \emptyset) \in R$ and thus $\mathcal{E}_1 \sim_{hp} \mathcal{E}_2$. \square

6 A logic with recursion: \mathcal{L}_{μ}

The logic \mathcal{L} discussed in the previous section is theoretically interesting as it allows to characterise logically the main truly concurrent equivalences. However, as a specification language, it has a limited expressiveness: even if one can “observe” events arbitrarily far in the future, a single formula in \mathcal{L} only describes properties where a finite number of events are executed. In order to overcome this limitation, in this section we study a fixpoint extension of the logic, where the use of recursion allows to express causal and concurrency properties of infinite computations. The resulting logic, denoted \mathcal{L}_{μ} , is a kind of first-order mu-calculus.

Let \mathcal{X}^a be a set of *abstract propositions*, ranged over by X, Y, \dots , that are intended to represent formulae possibly containing (unnamed) free event variables. Each abstract proposition has an arity $ar(X)$, which indicates the number of free event variables in X . An abstract proposition X can be turned into a formula by specifying a name for its free variables. For \mathbf{x} such that $|\mathbf{x}| = ar(X)$, we write $X(\mathbf{x})$ to indicate abstract proposition X whose free event variables are named \mathbf{x} . We call $X(\mathbf{x})$ a *proposition* and denote by \mathcal{X} the set of all propositions.

Definition 14 (syntax). *Let Var be a denumerable set of event variables and let \mathcal{X} be a set of propositions, as explained above. The syntax of \mathcal{L}_{μ} over the set of labels Λ is defined as follows:*

$$\varphi ::= X(\mathbf{x}) \mid \top \mid \varphi \wedge \varphi \mid \neg \varphi \mid (\mathbf{x}, \mathbf{y} < \mathbf{a} z) \varphi \mid \langle z \rangle \varphi \mid \mu X(\mathbf{x}).\varphi$$

where for formula $\mu X(\mathbf{x}).\varphi$, as usual, X must occur positively in φ and additionally, $fv(\varphi) = \mathbf{x}$.

Definition 15 (free variables). *The free variables of a formula φ in \mathcal{L}_μ are given as in Definition 9, with the addition of the following clauses:*

$$fv(X(\mathbf{x})) = \mathbf{x} \quad \text{and} \quad fv(\mu X(\mathbf{x}).\varphi) = \mathbf{x}.$$

In the following we will often use the set of free variables of a formula as a tuple. Thus it is convenient to assume that $fv(\cdot)$ returns a fixed tuple of variables. Note that the fact that variables \mathbf{x} are free in $X(\mathbf{x})$ and $\mu X(\mathbf{x}).\varphi$ is reflected in the definition of free variable substitution. For instance $X(\mathbf{x})[\mathbf{y}/\mathbf{x}] = X(\mathbf{y})$ and $(\mu X(\mathbf{x}).\varphi)[\mathbf{y}/\mathbf{x}] = \mu X(\mathbf{y}).(\varphi[\mathbf{y}/\mathbf{x}])$.

A least fixpoint operator μ has been added. In a recursive formula $\mu X(\mathbf{x}).\varphi$ the abstract proposition X can occur recursively in φ , possibly with a different tuple of variables which intuitively are to be used in the next iteration. As usual a greatest fixpoint operator can be encoded, by duality, as

$$\nu X(\mathbf{x}).\varphi = \neg(\mu X(\mathbf{x}).\neg\tilde{\varphi})$$

where $\tilde{\varphi}$ is the formula obtained replacing any occurrence of X in φ with $\neg X$ (in order to keep the positivity of the occurrences of X).

As an example, the existence of a run consisting of an infinite causal chain of **a**-actions can be expressed by the following formula:

$$\langle \mathbf{a} \mathbf{x} \rangle (\nu X(\mathbf{x}).\langle \mathbf{x} < \mathbf{a} \mathbf{y} \rangle X(\mathbf{y}))$$

The infinite causal chain is obtained by “passing” the event bound to y by the current execution to the next iteration so that it can be used as a cause in the corresponding execution. The execution outside the recursive formula binds x to an **a**-labelled event which will be the first in the causal chain.

In a fixpoint formula $\mu X(\mathbf{x}).\varphi$, the fixpoint operator binds all the free occurrences of the abstract proposition X in φ . This leads to the following notion of free abstract proposition.

Definition 16 (free propositions, substitution). *The set of free propositions in a formula φ in \mathcal{L}_μ , denoted $fp(\varphi)$, is defined inductively by*

$$\begin{aligned} fp(\top) &= \emptyset & fp(X(\mathbf{x})) &= \{X\} \\ fp(\varphi_1 \wedge \varphi_2) &= fp(\varphi_1) \cup fp(\varphi_2) \\ fp(\neg\varphi) &= fp(\langle \mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} \mathbf{z} \rangle \varphi) = fp(\langle \mathbf{z} \rangle \varphi) = fp(\varphi) \\ fp(\mu X(\mathbf{x}).\varphi) &= fp(\varphi) \setminus \{X\} \end{aligned}$$

Let φ be a formula in \mathcal{L}_μ . For an abstract proposition X and formula ψ such that $fv(\psi) = \mathbf{x}$, $|\mathbf{x}| = ar(X)$, we denote by $\varphi[\psi/X]$ the formula obtained from φ by replacing any free occurrence of $X(\mathbf{y})$ by $\psi[\mathbf{y}/\mathbf{x}]$.

A formula $\varphi \in \mathcal{L}_\mu$ is called *closed* when both $fv(\varphi)$ and $fp(\varphi)$ are empty.

Let us now move to the definition of the semantics. Legal pairs for a formula are defined exactly as in Definition 10. For instance the pair (C, η) is legal for the formula $X(\mathbf{x})$ if the set $C \cup \eta(\mathbf{x})$ is pairwise consistent. On the other hand, in addition to the (event variable) environment, the semantics of \mathcal{L}_μ also requires an interpretation for the propositions, mapping each proposition to a set of legal pairs.

Definition 17 (proposition environments). *Let \mathcal{E} be a PES. A proposition environment is a function $\pi : \mathcal{X} \rightarrow 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$ such that:*

1. $\pi(X(\mathbf{x})) \subseteq lp(X(\mathbf{x}))$ for any $X(\mathbf{x}) \in \mathcal{X}$, and
2. if $(C, \eta) \in \pi(X(\mathbf{x}))$ and $\eta'(\mathbf{y}) = \eta(\mathbf{x})$ pointwise, then $(C, \eta') \in \pi(X(\mathbf{y}))$.

We denote by $PEnv_{\mathcal{E}}$ the set of proposition environments, ranged over by π .

The first condition requires that the denotation for $X(\mathbf{x})$ only consists of legal pairs for $X(\mathbf{x})$. The second condition requires that the semantics of a proposition only depends on the events that the environment associates to its free variables and that it does not depend on the naming of the variables. Such a condition allows to extend Lemma 2 to the logic with recursion.

Updates of a proposition environment must be performed in order to maintain the validity of properties 1 and 2 above. For $\pi \in PEnv_{\mathcal{E}}$ and $S \subseteq lp(X(\mathbf{x}))$, we write $\pi[X(\mathbf{x}) \mapsto S]$ for the proposition environment defined by

$$\begin{aligned}\pi[X(\mathbf{x}) \mapsto S](X(\mathbf{y})) &= \{(C, \eta') \mid (C, \eta) \in S \wedge \eta'(\mathbf{y}) = \eta(\mathbf{x})\} \\ \pi[X(\mathbf{x}) \mapsto S](Y(\mathbf{y})) &= \pi(Y(\mathbf{y})) \quad \text{for } Y \neq X.\end{aligned}$$

Lemma 11. *Let \mathcal{E} be a PES, let $\varphi \in \mathcal{L}_{\mu}$ be a formula. and let $\mathbf{x} = fv(\varphi)$ be the tuple of free variables in φ .*

1. *If $(C, \eta) \in \llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}}$ and $\eta'(\mathbf{y}) = \eta(\mathbf{x})$ pointwise, then $(C, \eta') \in \llbracket \varphi[\mathbf{y}/\mathbf{x}] \rrbracket_{\pi}^{\mathcal{E}}$.*
2. *For any ψ it holds $\llbracket \psi[\varphi/X] \rrbracket_{\pi}^{\mathcal{E}} = \llbracket \psi \rrbracket_{\pi[X(\mathbf{x}) \mapsto \llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}}]}^{\mathcal{E}}$*

Proof. Both items can be proved by a routine induction (on φ for 1 and on ψ for 2). \square

In particular, from 1 above it follows that, as for \mathcal{L} , the semantics only depends on the events that the environment associates to the free variables of the formula, i.e., if $C \in \mathcal{C}(\mathcal{E})$ and η, η' are environments such that $\eta_{\mathbf{x}} = \eta'_{\mathbf{x}}$ then $(C, \eta) \in \llbracket \varphi \rrbracket^{\mathcal{E}}$ iff $(C, \eta') \in \llbracket \varphi \rrbracket^{\mathcal{E}}$.

Definition 18 (semantics). *Let \mathcal{E} be a PES. The denotation of a formula is given by the function*

$$\llbracket \cdot \rrbracket^{\mathcal{E}} : \mathcal{L}_{\mu} \rightarrow PEnv_{\mathcal{E}} \rightarrow 2^{\mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}}}$$

defined inductively as follows, where we write $\llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}}$ instead of $\llbracket \varphi \rrbracket^{\mathcal{E}}(\pi)$:

$$\begin{aligned}\llbracket \top \rrbracket_{\pi}^{\mathcal{E}} &= \mathcal{C}(\mathcal{E}) \times Env_{\mathcal{E}} \\ \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{\pi}^{\mathcal{E}} &= \llbracket \varphi_1 \rrbracket_{\pi}^{\mathcal{E}} \cap \llbracket \varphi_2 \rrbracket_{\pi}^{\mathcal{E}} \\ \llbracket \neg \varphi \rrbracket_{\pi}^{\mathcal{E}} &= lp(\varphi) \setminus \llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}} \\ \llbracket (\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z) \varphi \rrbracket_{\pi}^{\mathcal{E}} &= \{(C, \eta) \mid (C, \eta) \in lp((\mathbf{x}, \bar{\mathbf{y}} < \mathbf{a} z) \varphi) \text{ and} \\ &\quad \exists e \in E[C] \text{ such that} \\ &\quad \lambda(e) = \mathbf{a} \wedge \eta(\mathbf{x}) < e \wedge \eta(\mathbf{y}) \parallel e \\ &\quad \wedge (C, \eta[z \mapsto e]) \in \llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}}\} \\ \llbracket \langle z \rangle \varphi \rrbracket_{\pi}^{\mathcal{E}} &= \{(C, \eta) \mid C \xrightarrow{\eta(z)} C' \wedge (C', \eta) \in \llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}}\} \\ \llbracket X(\mathbf{x}) \rrbracket_{\pi}^{\mathcal{E}} &= \pi(X(\mathbf{x})) \\ \llbracket \mu X(\mathbf{x}).\varphi \rrbracket_{\pi}^{\mathcal{E}} &= lfp(f)\end{aligned}$$

where $lfp(f)$ is the least fixed point of the function $f : 2^{lp(X(\mathbf{x}))} \rightarrow 2^{lp(X(\mathbf{x}))}$ that maps $S \subseteq lp(X(\mathbf{x}))$ into

$$f(S) = \llbracket \varphi \rrbracket_{\pi[X(\mathbf{x}) \mapsto S]}^{\mathcal{E}}$$

When $(C, \eta) \in \llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}}$ we say that the PES \mathcal{E} satisfies the formula φ in the configuration C and environments η, π and write $\mathcal{E}, C \models_{\eta, \pi} \varphi$. For closed formulae φ , we write $\mathcal{E} \models \varphi$, when $\mathcal{E}, \emptyset \models_{\emptyset, \emptyset} \varphi$.

It can be easily proved that Lemma 1 extends to \mathcal{L}_{μ} , i.e., for any formula $\varphi \in \mathcal{L}_{\mu}$, its denotation only contains legal pairs, that is $\llbracket \varphi \rrbracket_{\pi}^{\mathcal{E}} \subseteq lp_{\mathcal{E}}(\varphi)$. Note also that the semantics of recursive formulae is well-given. In fact, $\pi[X(\mathbf{x}) \mapsto S]$ is a well-defined proposition environment, since $S \subseteq lp(X(\mathbf{x}))$. Moreover $f(S) = \llbracket \varphi \rrbracket_{\pi[X(\mathbf{x}) \mapsto S]}^{\mathcal{E}} \subseteq lp(\varphi)$ by the previous observation, and $lp(X(\mathbf{x})) = lp(\varphi)$ since

$fv(\varphi) = \mathbf{x}$ by definition of the syntax of \mathcal{L}_μ . Therefore, correctly, $f(S) \subseteq lp(X(\mathbf{x}))$. Moreover, the least fixed point of f exists by Knaster-Tarski theorem since the set $2^{lp(X(\mathbf{x}))}$ ordered by subset inclusion is a complete lattice and the function f used in the definition is monotone. This can be easily checked by inspection of the definition of the semantics (Definition 18), keeping in mind that $X(\mathbf{x})$ is required to occur positively in φ .

As for the non-recursive fragment \mathcal{L} , the logic \mathcal{L}_μ could be defined in positive form. The corresponding syntax, given below, includes the dual operators and omits negation, which can then be encoded by duality.

$$\begin{aligned} \varphi ::= & X(\mathbf{x}) \mid \top \mid \varphi \wedge \varphi \mid (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi \mid \langle z \rangle \varphi \mid \mu X(\mathbf{x}).\varphi \\ & \text{F} \mid \varphi \vee \varphi \mid \{\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z\} \varphi \mid [z] \varphi \mid \nu X(\mathbf{x}).\varphi \end{aligned}$$

In the following we will freely use the dual operators.

6.1 Examples

In the previous section we observed that standard HM logic can be viewed as a fragment of \mathcal{L} where we only use the (derived) modality $\langle \mathbf{a} x \rangle$. Similarly, the propositional μ -calculus corresponds to a fragment of the the general logic \mathcal{L}_μ where we avoid references to causally dependent/independent events. In particular, since in recursive formulae we do not express causal links between event (variables) used in different iterations, we can use only propositions without free variables (i.e., of arity 0). Therefore, the μ -calculus corresponds to the following fragment of \mathcal{L}_μ :

$$\varphi ::= X(\epsilon) \mid \top \mid \varphi \wedge \varphi \mid \neg \varphi \mid (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} z) \varphi \mid \langle z \rangle \varphi \mid \mu X(\epsilon).\varphi$$

For simplicity in the following we omit trailing empty tuples of variables, writing X instead of $X(\epsilon)$.

As first examples of \mathcal{L}_μ formulae we thus have some basic safety and liveness properties inherited from the μ -calculus. For a fixed closed formula ψ , representing a property of interest:

- ψ holds in every reachable state
 $Inv(\psi) = \nu X. (\psi \wedge \llbracket _ z \rrbracket X)$;
- ψ eventually holds in some state
 $Pos(\psi) = \mu X. (\psi \vee \langle _ z \rangle X)$;
- there is a complete (finite terminated or infinite) computation where ψ always holds
 $Safe(\psi) = \nu X. (\psi \wedge (\llbracket _ z \rrbracket \text{F} \vee \langle _ x \rangle X))$;
- in every complete computation eventually ψ holds
 $Ev(\psi) = \mu X. (\psi \vee (\langle _ z \rangle \top \wedge \llbracket _ x \rrbracket X))$.

When moving to the full logic, property ψ can include concurrency and causal features. In case ψ is not closed, denoted by \mathbf{x} the tuple of free variables in ψ , in order to respect the syntax any occurrence of X above must be replaced by $X(\mathbf{x})$. Then we can consider $Ev(\langle \mathbf{a} z \rangle \otimes \langle \mathbf{a} z' \rangle)$ saying that eventually there will be a concurrent step consisting of two events, labelled \mathbf{a} and \mathbf{b} , respectively, or $Inv(\langle \mathbf{r} z \rangle Ev(\langle \mathbf{z} < \mathbf{s} z' \rangle))$ saying that any \mathbf{r} -labelled event will be eventually followed by a \mathbf{s} -labelled event caused by it (e.g., any request will be eventually served).

More generally, logic \mathcal{L}_μ allows one to express causal and concurrency properties of infinite computation, where events occurring in different fixpoint iterations are possibly related. We next provide a number of further examples.

- There is a causal chain of \mathbf{b} -labelled events reaching a state where \mathbf{a} can be fired:

$$\langle \mathbf{a} y \rangle \top \vee \langle \mathbf{b} x \rangle (\mu X(x). (\langle \mathbf{a} z \rangle \top \vee \langle x < \mathbf{b} y \rangle X(y)))$$

- There is an executable \mathbf{a} -labelled event such that in every configuration reached by executing events which are concurrent with it, a \mathbf{c} -labelled event can be executed:

$$(\mathbf{a} x) (\langle x \rangle \top \wedge \nu X(x). (\langle \mathbf{c} z \rangle \top \wedge \llbracket \overline{\mathbf{x}} < _ y \rrbracket X(x)))$$

- It is always possible to perform a step consisting of two concurrent events labelled by **a** and **b**, after executing any number of events labelled **c**:

$$\nu X. ((\langle \mathbf{a} \, z \rangle \otimes \langle \mathbf{b} \, z' \rangle) \mathsf{T} \wedge \llbracket \mathbf{c} \, w \rrbracket X)$$

- There is a finite sequence of (not necessarily related) steps consisting of two events labelled by **a** and **b**, respectively, and finally a **c**-labelled event:

$$\mu X. (\langle \mathbf{c} \, z \rangle \mathsf{T} \vee (\langle \mathbf{a} \, z \rangle \otimes \langle \mathbf{b} \, z' \rangle) X)$$

6.2 Invariance of logical equivalence

We show that the addition of fixpoints formulae does not alter the logical equivalence, that still coincides with hhp-bisimilarity, i.e., $\equiv_{\mathcal{L}} = \equiv_{\mathcal{L}_\mu} = \sim_{hhp}$. (Recall that in the paper we are limiting ourselves to image-finite PESs.) For this aim we introduce an infinitary version of the logic \mathcal{L}_μ , which can be exploited to define fixpoint approximants (see, e.g., [BS06]).

Let us denote by \mathcal{L}_μ^∞ an extension of \mathcal{L}_μ with infinite conjunctions, i.e., formulae of \mathcal{L}_μ^∞ are defined by the grammar

$$\varphi ::= X(\mathbf{x}) \mid \mathsf{T} \mid \bigwedge_{i \in I} \varphi_i \mid \neg \varphi \mid (\mathbf{x}, \overline{\mathbf{y}} < \mathbf{a} \, z) \varphi \mid \langle z \rangle \varphi \mid \mu X(\mathbf{x}).\varphi$$

The semantics of \mathcal{L}_μ^∞ is defined as in Definition 18, replacing the clause for conjunction with $\llbracket \bigwedge_{i \in I} \varphi_i \rrbracket_\pi^\mathcal{E} = \bigcap_{i \in I} \llbracket \varphi_i \rrbracket_\pi^\mathcal{E}$. We denote by \mathcal{L}^∞ the fragment of \mathcal{L}_μ^∞ not including propositions and fixpoint operators.

Definition 19 (approximants). *The α -th approximant of a fixpoint formula in \mathcal{L}_μ^∞ , for an ordinal α , is a formula in \mathcal{L}^∞ , inductively defined as follows:*

$$\begin{aligned} \mu^0 X(\mathbf{x}).\varphi &= \mathsf{F} \\ \mu^{\alpha+1} X(\mathbf{x}).\varphi &= \varphi[\mu^\alpha X(\mathbf{x}).\varphi/X] \\ \mu^\lambda X(\mathbf{x}).\varphi &= \bigvee_{\alpha < \lambda} \mu^\alpha X(\mathbf{x}).\varphi \quad \text{for } \lambda \text{ a limit ordinal} \end{aligned}$$

A fixpoint formula $\mu X(\mathbf{x}).\varphi$ is intuitively equivalent to the (infinite) disjunction of its approximants. More formally:

Lemma 12 (fixpoint unfolding via approximants). *Let \mathcal{E} be a PES. For any formula $\mu X(\mathbf{x}).\varphi$ in \mathcal{L}_μ^∞ there exists an ordinal α such that*

$$\llbracket \mu X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E} = \llbracket \mu^\alpha X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E}.$$

Proof. Recall that $\llbracket \mu X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E} = \text{lfp}(f)$ where $f : 2^{lp(X(\mathbf{x}))} \rightarrow 2^{lp(X(\mathbf{x}))}$ is the function defined by $f(S) = \llbracket \varphi \rrbracket_{\pi[X(\mathbf{x}) \mapsto S]}^\mathcal{E}$.

We already noted that the function f is monotone in $2^{lp(X(\mathbf{x}))}$ ordered by subset inclusion. Hence its least fixpoint can be obtained by iterating f on \emptyset , the bottom element of the lattice, i.e., there exists an ordinal α such that $\text{lfp}(f) = f^\alpha(\emptyset)$, where $f^0(\emptyset) = \emptyset$, $f^{\alpha+1}(\emptyset) = f(f^\alpha(\emptyset))$ and $f^\lambda(\emptyset) = \bigcup_{\alpha < \lambda} f^\alpha(\emptyset)$ for λ a limit ordinal.

The observation that for any ordinal α it holds that $f^\alpha(\emptyset) = \llbracket \mu^\alpha X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E}$ allows us to conclude. The latter can be proved by transfinite induction on α .

$$(\alpha = 0) \quad \llbracket \mu^0 X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E} = \llbracket \mathsf{F} \rrbracket_\pi^\mathcal{E} = \emptyset = f^0(\emptyset)$$

$(\alpha \rightarrow \alpha + 1)$ We have that

$$\begin{aligned} \llbracket \mu^{\alpha+1} X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E} &= && [\text{def. of } \mu^{\alpha+1} X(\mathbf{x}).\varphi] \\ &= \llbracket \varphi[\mu^\alpha X(\mathbf{x}).\varphi/X] \rrbracket_\pi^\mathcal{E} && [\text{Lemma 11}] \\ &= \llbracket \varphi \rrbracket_{\pi[X(\mathbf{x}) \mapsto \llbracket \mu^\alpha X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E}]}^\mathcal{E} && [\text{def. of } f] \\ &= f(\llbracket \mu^\alpha X(\mathbf{x}).\varphi \rrbracket_\pi^\mathcal{E}) && [\text{inductive hypothesis}] \\ &= f(f^\alpha(\emptyset)) \end{aligned}$$

(λ limit ordinal) We have

$$\begin{aligned}
\{\mu^\lambda X(\mathbf{x}).\varphi\}_\pi^\mathcal{E} &= \\
&= \{\bigvee_{\alpha < \lambda} \mu^\alpha X(\mathbf{x}).\varphi\}_\pi^\mathcal{E} = \\
&= \bigcup_{\alpha < \lambda} \{\mu^\alpha X(\mathbf{x}).\varphi\}_\pi^\mathcal{E} = \\
&= \bigcup_{\alpha < \lambda} f^\alpha(\emptyset) = \\
&= f^\lambda(\emptyset)
\end{aligned}$$

□

We can finally prove that the logical equivalences induced by \mathcal{L} and \mathcal{L}_μ are the same and they both coincide with \sim_{hhp} .

Corollary 1 (invariance of logical equivalence). *The logical equivalences of \mathcal{L} and \mathcal{L}_μ coincide with \sim_{hhp} .*

Proof. First of all, since \mathcal{L}_μ extends \mathcal{L} , clearly $\equiv_{\mathcal{L}_\mu}$ implies $\equiv_{\mathcal{L}}$ which in turn, by Proposition 1, implies \sim_{hhp} . Hence $\equiv_{\mathcal{L}_\mu}$ implies \sim_{hhp} . For the opposite direction, note that Proposition 2 can be straightforwardly adapted to logic \mathcal{L}^∞ (as finiteness of conjunction plays no role in the proof). Hence \sim_{hhp} implies $\equiv_{\mathcal{L}^\infty}$. An inductive argument, using Lemma 12, allows one to show that for any closed formula in \mathcal{L}_μ^∞ (and thus in particular any formula in \mathcal{L}_μ), there exists an equivalent formula in \mathcal{L}^∞ , obtained by replacing all fixpoint operators with suitable approximants. Therefore $\equiv_{\mathcal{L}^\infty}$ implies $\equiv_{\mathcal{L}_\mu}$, hence \sim_{hhp} implies $\equiv_{\mathcal{L}_\mu}$ as desired. □

7 Conclusions: related and future work

We have introduced a logic for true concurrency, which allows to predicate on events in computation and their mutual dependencies (causality and concurrency). The logic subsumes standard HM logic and provides a characterisation of the most widely known true concurrent behavioural equivalences: hhp-bisimilarity is the logical equivalence induced by the full logic, and suitable fragments are identified which induce hp-bisimilarity, pomset and step bisimilarity.

As we mentioned in the introduction, there is a vast literature relating logical and operational views of true concurrency, however, to the best of our knowledge, a uniform logical counterpart of the true concurrent spectrum was still missing. An exhaustive account of the related literature is impossible; we just recall here the approaches that most closely relate to our work.

In [DNF90,PLS94,Che92] the causal structure of concurrent systems is pushed into the logic. The paper [DNF90] considers modalities which describe pomset transitions, thus providing an immediate characterisation of pomset bisimilarity. Moreover, [DNF90,PLS94,Che92] show that by tracing the history of states and adding the possibility of reverting pomset transitions, one obtains an equivalence coarser than hp-bisimilarity and incomparable with pomset bisimilarity, called weak hp-bisimilarity. Our logic intends to be more general by also capturing the interplay between concurrency and branching, which is not observable at the level of hp-bisimilarity.

A recent work [GB09,Gut09] introduces a fixpoint modal logic for true concurrent models, called Separation Fixpoint Logics (SFL). This includes modalities which specify the execution of an action causally dependent/independent on the last executed one. Moreover, a “separation operator” deals with concurrently enabled actions. The approach of [GB09,Gut09] is inspired by the so-called *Independence-Friendly Modal Logic* (IFML) [BF02], which includes a modality that allows to specify that the current executed action is independent from a number of previously executed ones. In this sense IFML is similar in spirit to our logic. The equivalences induced by fragments of IFML are not standard in the true concurrent spectrum. (The fragment of the logic without the separation operator captures an equivalence referred to as hp-bisimilarity, but which actually is a weakening of it [Frö10]. For similar reasons, the full logic induces an equivalence which is weaker than hhp-bisimilarity, and incomparable with hp-bisimilarity). Still a deeper comparison with this approach represents an interesting open issue.

Several classical papers have considered temporal logics with modalities corresponding to the “retraction” or “backward” execution of computations. In particular [NC95, Bed91, HS85] study a so-called path logic with a future perfect (also called past tense) modality: $@a\varphi$ is true when φ holds in a state which can reach the current one with an a -transition. When interpreted over transition systems with independence, in absence of autoconcurrency, the logic characterises hhp-bisimilarity. In [NC95] it is shown that, taking autoconcurrency into account, the result can be extended at the price of complicating the logic (roughly, the logic needs an operator to undo a specific action performed in the past).

Compared to these works, the main novelty of our approach resides in the fact that the logic \mathcal{L} provides a characterisation of the different equivalences in a simple, unitary logical framework. In order to enforce this view, we intend to pursue a formal comparison with the logics for concurrency introduced in the literature. It is easy to see that the execution modalities of [GB09, Gut09] can be encoded in \mathcal{L} since they only refer to the last executed event, while the formulae in \mathcal{L} can refer to any event executed in the past. On the other hand, the “separation operator” of [GB09, Gut09], as well as the backward modalities mentioned above (past tense, future perfect, reverting pomset transitions) are not immediately encodable in \mathcal{L} . A deeper investigation would be of great help in shading further light on the truly concurrent spectrum. Moreover \mathcal{L} suggests an alternative, forward-only, operational definition of hhp-bisimulation, which could be inspiring also for other reverse bisimulations [PU10].

Interestingly, the idea of considering a logic with event variables is taken also in a very recent work [PU11], which provides an elegant characterisation of (h)hp-bisimilarity via a logic, called *Event Identifier Logic* (EIL), with backward execution modality. The logic includes three operators: $\langle x:a \rangle$, $(x:a)$ and $\langle\langle x \rangle\rangle$. The formula $\langle x:a \rangle\varphi$ holds when an a -labelled event can be executed and then φ holds. The formula $(x:a)\varphi$ holds in if the current state (configuration) there is an a -labelled event and then φ holds. In both cases, the event is bound to variable x for future references. Finally, $\langle\langle x \rangle\rangle$ holds when the event bound to x can be undone and then φ holds. The reason why both logics capture hhp-bisimilarity is conceptually clear: the possibility of performing backward steps can be seen as a mean of exploring alternative different futures. The very same possibility is “primitive” in our logic where we can explore the future of a configuration, without executing the corresponding events. However, the formal relationships between EIL and our logic (e.g., the possibility of encoding backward steps in our logic) is still to be understood and represents a stimulating direction of future research.

As a byproduct of such an investigation, we foresee the identification of interesting extensions of the concurrent spectrum, both at the logical and at the operational side. For instance, it can be shown that the fragment of \mathcal{L} where the operator $(x, \bar{y} < a z)$ is restricted to bind z to events consistent with the those already quantified induces an equivalence which admits a natural operational definition, is still decidable and lies in between hp- and hhp-bisimilarity, still being different from the equivalences in [GB09, Gut09].

Connected to this, model-checking and decidability issues are challenging directions of future investigation (see [Pen95] for a survey of these issues over partial order temporal logics and logics based on event structures having explicit operators representing concurrency, causality and conflict). It is known that hhp-bisimilarity is undecidable, even for finite state systems [JNS03], while hp-bisimilarity is decidable. Characterising decidable fragments of the logic could be helpful in drawing a clearer separation line between decidability and undecidability of concurrent equivalences. A promising direction is to impose a bound on the “causal depth” of the future which the logic can quantify on. In this way one gets a chain of equivalences, coarser than hhp-bisimilarity, which should be closely related with n -hhp bisimilarities introduced and shown to be decidable in [FH99]. As for verification, we aim at investigating the automata-theoretic counterpart of the logic. In previous papers, hp-bisimilarity has been characterised in automata-theoretic terms using HD-automata [MP97] or Petri nets [Vog91]. It seems that HD-automata [MP97] could provide a suitable automata counterpart of the fragment \mathcal{L}_{hp} . Also the game-theoretical approach proposed in [GB09, Gut09] for the fixpoint separation logic can be a source of inspiration.

Just note that the model checking problem is not trivial since a formula can have only infinite models, even if we limit ourselves to the finite fragment. For instance, the formula $\langle a w \rangle T \wedge$

$\neg(a x) \neg(x < a y) \top$ only holds in an PES which contains an infinite causal chain of a -labelled event. Preliminary investigations lead us to conjecture model-checking to be decidable on finite state systems for the fixpoint extension of \mathcal{L}_{hp} , \mathcal{L}_p and \mathcal{L}_s .

Acknowledgements. We are grateful to Luca Aceto, Sibylle Fröschle and to the anonymous reviewers for their inspiring comments on the conference version of the paper.

References

- Bed91. M. A. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical report, Polish Academy of Sciences, 1991.
- BF02. J. Bradfield and S. Fröschle. Independence-friendly modal logic and true concurrency. *Nordic Journal of Computing*, 9(1):102–117, 2002.
- BK05. J. Bradfield and K. Kreutzer. The complexity of independence-friendly fixpoint logic. In C.-H. L. Ong, editor, *Proceedings of CLS’05*, volume 3634 of *LNCS*, pages 355–368. Springer, 2005.
- BS06. Julian Bradfield and Colin Stirling. Modal mu-calculi. In P. Blackburn, van Benthem J., and F. Wolter, editors, *Handbook of Modal Logic*, pages 721–756. Elsevier, 2006.
- Che92. F. Chierief. Back and forth bisimulations on prime event structures. In D. Etienne and J.-C. Syre, editors, *Proceedings of PARLE’92*, volume 605 of *LNCS*, pages 843–858. Springer, 1992.
- Dam96. M. Dam. Model checking mobile processes. *Information and Computation*, 129(1):35–51, 1996.
- DFG98. M. Dam, L.-Å. Fredlund, and D. Gurov. Toward parametric verification of open distributed systems. In W. P. de Roever, H. Langmaack, and A. Pnueli, editors, *Proceedings of COMPOS’97*, volume 1536 of *LNCS*, pages 150–185. Springer, 1998.
- DNF90. R. De Nicola and G. Ferrari. Observational logics and concurrency models. In K. V. Nori and C. E. V. Madhavan, editors, *Proceedings of FST-TCS’90*, volume 472 of *LNCS*, pages 301–315. Springer, 1990.
- FH99. S. Fröschle and T. Hildebrandt. On plain and hereditary history-preserving bisimulation. In M. Kutylowski, L. Pacholski, and T. Wierzbicki, editors, *Proceedings of MFCS’99*, volume 1672 of *LNCS*, pages 354–365. Springer, 1999.
- Frö10. S. Fröschle. Personal communication, 2010.
- GB09. J. Gutierrez and J. C. Bradfield. Model-checking games for fixpoint logics with partial order models. In M. Bravetti and G. Zavattaro, editors, *Proceedings of CONCUR’09*, volume 5710 of *LNCS*, pages 354–368. Springer, 2009.
- Gut09. J. Gutierrez. Logics and bisimulation games for concurrency, causality and conflict. In L. de Alfaro, editor, *Proceedings of FoSSaCS’09*, volume 5504 of *LNCS*, pages 48–62. Springer, 2009.
- HM85. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.
- HS85. M. Hennessy and C. Stirling. The power of the future perfect in program logics. *Information and Control*, 67(1-3):23–52, 1985.
- JNS03. M. Jurdzinski, M. Nielsen, and J. Srba. Undecidability of domino games and hhp-bisimilarity. *Information and Computation*, 184(2):343–368, 2003.
- MP97. U. Montanari and M. Pistore. Minimal transition systems for history-preserving bisimulation. In R. Reischuk and M. Morvan, editors, *Proceedings of STACS’97*, volume 1200 of *LNCS*, pages 413–425. Springer, 1997.
- NC95. M. Nielsen and C. Clausen. Games and logics for a noninterleaving bisimulation. *Nordic Journal of Computing*, 2(2):221–249, 1995.
- Pen95. W. Penczek. Branching time and partial order in temporal logics. In *Time and Logic: A Computational Approach*, pages 179–228. UCL Press, 1995.
- PLS94. S. Pinchinat, F. Laroussinie, and Ph. Schnoebelen. Logical characterization of truly concurrent bisimulation. Technical Report 114, LIFIA-IMAG, Grenoble, 1994.
- PU10. I. Phillips and I. Ulidowski. Reverse bisimulations on stable configuration structures. In B. Klin and P. Sobociński, editors, *Proc. of SOS’09*, volume 18 of *Electronic Proceedings in Theoretical Computer Science*, pages 62–76, 2010.
- PU11. I. Phillips and I. Ulidowski. A logic with reverse modalities for history-preserving bisimulations. In B. Luttik and F. Valencia, editors, *Proceedings of EXPRESS 2011*, volume 64 of *EPTCS*, pages 104–118, 2011.

- vG01. R.J. van Glabbeek. The linear time – branching time spectrum I; the semantics of concrete, sequential processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 1, pages 3–99. Elsevier, 2001.
- vGG01. R.J. van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37(4/5):229–327, 2001.
- Vog91. W. Vogler. Deciding history preserving bisimilarity. In J. Leach Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proceedings of ICALP’91*, volume 510 of *LNCS*, pages 495–505. Springer, 1991.
- Win87. G. Winskel. Event Structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.
- WN95. G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of logic in Computer Science*, volume 4. Clarendon Press, 1995.